UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Urb. Montalbán – La Vega- Apartado 29086

Teléfono: +58212 - 4074407 Fax: +58212 - 4074447

Caracas, 1021 –Venezuela

Facultad de Ingeniería

Escuela de Ingeniería Informática

Sistema de tareas y consulta de tráfico por posicionamiento global para dispositivos móviles bajo la plataforma Android de Google

TRABAJO ESPECIAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar por el título de

INGENIERO EN INFORMÁTICA

REALIZADO POR: Julián Alberto Izquierdo Ayala

Rubén Antonio López García

PROFESOR GUÍA: Ing. Ricardo Casanova

FECHA: 25 de septiembre de 2009

ÍNDICE DE CONTENIDO

SINOPSIS....8

CAPÍTULO I....9

INTRODUCCIÓN....9

Planteamiento del problema....9

Objetivo general....10

Objetivos específicos....10

Justificación....11

Limitaciones y alcances....14

CAPÍTULO II....17

MARCO DE REFERENCIA / TEÓRICO....17

Tecnología GPS....17

Ubicación al alcance de todos....18

Tecnología Android....19

Android Software Development Kit (SDK)....21

Empezando el desarrollo: Eclipse y Android....22

Envío y recepción de mensajes de texto....23

CAPÍTULO III....25

MARCO METODOLÓGICO....25

CAPÍTULO IV....28

DESARROLLO....28

Selección de las herramientas de desarrollo....28

Integración de la aplicación con Google Maps....32
Módulo de Registro de Coordenadas....35
Módulo de Gestión de Tareas....40
Módulo de Gestión de Alarmas Simples....40
Módulo de Gestión de Recordatorios....46
Módulo de Gestión de Alarmas SMS....51
Módulo de Gestión de Alarmas Email....56
Módulo de Consulta de Tráfico....60
Módulo de Servidor de Mensajes de Texto....66
Android frente a otras plataformas....72
Estudio comparativo....72
CAPÍTULO V....75
RESULTADOS....76
CONCLUSIONES....78
RECOMENDACIONES....79
REFERENCIAS BIBLIOGRÁFICAS....80

ÍNDICE DE FIGURAS

Figura 1. Arquitectura Android.....20

Figura 2. Escritorio del Emulador que ofrece la plataforma Android.....21

Figura 3. Proyecto nuevo del tipo Android en Eclipse.....23

Figura 4. Modelo Cascada con Reducción de riesgos.....27

Figura 5. Dispositivo HTC Dream G1 Android de Google.....29

ÍNDICE DE TABLAS

Tabla 1. Características conocidas del dispositivo móvil HTC Dream G1 Android de Google.....30

Tabla 2. Especificaciones del equipo móvil Motorola C650.....31

Tabla 3. Campos de la tabla "Lugar" de la base de datos asociada a la aplicación a desarrollar.....37

Tabla 4. Campos de la tabla "Alarma Normal" de la base de datos asociada a la aplicación a desarrollar.....41

Tabla 5. Campos de la tabla "Recordatorio" de la base de datos asociada a la aplicación a desarrollar.....47

Tabla 6. Campos de la tabla "Alarma Mensaje" de la base de datos asociada a la aplicación a desarrollar.....52

Tabla 7. Campos de la tabla "Alarma Correo" de la base de datos asociada a la aplicación a desarrollar.....57

Tabla 8. Tablas de la base de datos asociada al Módulo de Consulta de Tráfico.....62

Tabla 9. Tablas de la base de datos asociada al Módulo de Servidor de mensajes de texto.....67

Tabla 10. Funcionalidades y oportunidades de desarrollo: LiMo, Symbian, OS X Mobile y Android.....74

SINOPSIS

El presente trabajo se centra principalmente en dar a conocer las facilidades que ofrece la plataforma Android, como sistema operativo para dispositivos móviles. A través de esta nueva tecnología en ascenso, se pudieron conocer todas las oportunidades y funcionalidades que presente logrando integrándolas en una sola aplicación, que resumiera el uso de un dispositivo de posicionamiento global y la configuración de alarmas en un sólo sistema: el Sistema de tareas y consulta de tráfico por posicionamiento global.

Durante el desarrollo de este trabajo se pudo comprender a cabalidad todas las herramientas que ofrece la tecnología Android, incluso pudiendo llegar a conocer tecnologías similares, logrando conocer sus ventajas y desventajas sobre ellas.

No sólo se logró integrar funcionalidades de la tecnología, sino solucionar necesidades reales de los usuarios de dispositivos móviles con estas oportunidades: configurar alarmas y alertas alrededor de un lugar de preferencia del usuario, envío de mensajes de texto y correos electrónicos de una manera automática, consultas acerca del tráfico en rutas muy transitadas en la ciudad de Caracas, y muchas otras operaciones a nivel gráfico que aproveche la funcionalidad del dispositivo GPS del teléfono celular.

En este Trabajo de Grado se encontraron muchas limitaciones, restricciones y obstáculos que interfirieron en el desarrollo de la solución final. Sin embargo, Android dotó siempre al equipo de las herramientas necesarias para seguir adelante, apoyado de las decenas de comunidades de desarrolladores que como nosotros, se toparon con la curiosidad de adentrarse en el mundo de esta novedosa tecnología.

CAPÍTULO I
INTRODUCCIÓN

Planteamiento del problema

Actualmente se encuentra en auge el uso de dispositivos de posicionamiento global, o GPS, con los cuales las personas pueden ubicarse en cualquier lugar o ir adonde quiera con un guía virtual que está configurado para llevar a cabo ese trabajo. Asimismo, el alcance de esta tecnología se ha ido expandiendo poco a poco hasta llegar a los teléfonos celulares, los cuales están dotados de mapas que permiten ubicar a los usuarios desde sus equipos móviles.

Sin embargo, estas funcionalidades son limitadas. Generalmente ofrecen funciones de posicionamiento sin ir más allá, dejando de lado un sinfín de funciones que se podrían implementar tomando en cuenta la ubicación del dispositivo según el lugar del mundo en el que se encuentre.

Los usuarios de dispositivos móviles suelen utilizar estas funcionalidades casualmente para saber en qué lugar se encuentran de la ciudad. Sin embargo, suelen frecuentar algunos lugares más que otros (sus hogares, lugares de trabajo, etc.), y asociados a estos lugares, para los usuarios, se desprenden otras necesidades: requieren del envío de mensajes a sus familiares cuando llegan a un lugar, necesitan recordar ciertas cosas al llegar a su oficina, o antes de llegar; también que al estar en el hogar se les recuerde periódicamente cierta información, o incluso si requieren poder avisar a alguien al dejar su lugar de trabajo.

Otro factor que se podría considerar es el tráfico en Caracas, el cual la mayoría de las veces es sumamente fuerte. Ante esto, surge la necesidad de que el usuario pueda saber, antes de salir a un lugar, que tan congestionada se encuentra la ruta que tomará, y de esta manera poder tomar una decisión de qué ruta tomar o planificar de mejor manera su tiempo en el trayecto desde su punto de salida hasta su destino.

Esto se podría solucionar con la implementación de una solución que ayude a que los usuarios aprovechen al máximo su dispositivo de posicionamiento, permitiendo que con él puedan abarcar más eventos y consultas que le faciliten recordar algo a sí mismo o a las personas de su preferencia, o conocer previamente lo congestionada que está una ruta transitada en Caracas.

Objetivo General

Desarrollar una aplicación sobre la nueva plataforma para dispositivos móviles Android de Google que por medio del sistema de posicionamiento (GPS) del teléfono celular (HTC G1) permita disparar alarmas relacionadas a tareas, recordatorios y eventos, y además permita al usuario consultar el tiempo aproximado que tardará en llegar de un punto a otro por diferentes rutas, en la ciudad de Caracas.

Objetivos específicos

- Evaluar y seleccionar las herramientas de desarrollo a utilizar, mediante el estudio de la tecnología, las funcionalidades disponibles y su integración al momento de programar.
- Desarrollar una aplicación que, interactuando con el GPS del dispositivo celular, presente un sistema de tareas y consulta de rutas de tráfico en Caracas, integrando el uso de alarmas, recordatorios, uso de mensajería instantánea y correo electrónico.
- Integrar el sistema a ser desarrollado con la aplicación de mapas de Google, Google Maps.
- Desarrollar la estructura de almacenamiento del registro de coordenadas de lugares de preferencia por parte de los usuarios, con la información contenida en los mapas y los datos ingresados por los usuarios.
- Desarrollar e implantar un servidor de mensajes de texto, el cual recibirá mensajes con datos (hora, fecha y nombre) acerca del tráfico en las rutas más transitadas de Caracas cuando un usuario de la aplicación entre en esas vías. El servidor enviará mensajes como respuestas a consultas que realicen los usuarios sobre el tráfico.

- Desarrollar e implantar un sistema que interactúe con una base de datos que permita el almacenamiento de la información que llegue al servidor de mensajes de texto. Con este sistema, el usuario tendrá acceso a información relevante con respecto a las rutas disponibles, rutas óptimas, así como la generación de reportes estadísticos relacionados con esas rutas según el tiempo.
- Realizar un estudio que presente una comparación entre la aplicación a desarrollar con otras aplicaciones análogas que existan en el mercado, sobre otras plataformas similares a Android. El estudio se hará con el fin de descubrir las posibles ventajas e inconvenientes de la aplicación a desarrollar con respecto a otras aplicaciones implementadas en otros ambientes de desarrollo.

Justificación

Con el desarrollo de esta aplicación se pretende explorar el desarrollo de una nueva aplicación para teléfonos celulares bajo la plataforma Android, lanzada por la empresa Google en noviembre de 2007. Se pretende extender el uso de los dispositivos de posicionamiento (GPS) a nivel de telefonía celular y ofrecer una herramienta que pueda ayudar a los usuarios a realizar varias operaciones simultáneamente y de manera automatizada.

Con el GPS, el usuario puede estar al tanto de dónde está y a dónde ir. En cambio, con la implantación de una aplicación de tareas, no sólo el usuario podrá saber dónde se encuentra y las posibilidades de a dónde podrá dirigirse, sino con ello podrá también identificar lugares a los cuales usualmente se dirige, así como, el disparo de eventos de su preferencia que irán desde simples recordatorios, hasta consultas sobre el tráfico en Caracas.

No sólo se podrán asociar tareas a los lugares para los que sean configuradas, sino también el usuario podrá asociarlas con respecto al tiempo que éste decida, e incluso asociarlas a un lugar y a un tiempo a la misma vez, ofreciendo un alcance mayor para sus necesidades. Con esto, el usuario puede hacerse de una agenda de tareas completa para, además de ayudarlo a recordar a él mismo asuntos que sean de su conveniencia, pueda

también poder hacer recordatorios a otros usuarios mediante mensajes de texto desde su propio celular.

Por ejemplo, poder configurar el teléfono celular para que envíe un mensaje a los padres cuando el hijo llegue a algún sitio, recordar en casa periódicamente que hay que sacar la basura, recordar cumpleaños y otros eventos importantes cada cierto tiempo o al llegar a un lugar, e infinidad de combinaciones posibles entre estas opciones.

Adicionalmente, también el sistema de posicionamiento se puede aprovechar desde un mayor enfoque. Hoy en día, el tráfico en la ciudad de Caracas se ha vuelto bastante engorroso para la mayoría de los conductores. No se sabe qué rutas tomar de ida para el trabajo, el hogar, u otro lugar de frecuente visita. Aunque hay guías que ayudan con esto, no son muy fiables en algunos casos. Es por eso que el sistema de posicionamiento se podría aprovechar. Se creará un servidor de mensajes de texto que guarde datos en cuanto a hora, fecha y nombre entre dos lugares. Estos datos serán enviados periódicamente cada vez que un usuario de la aplicación sobre Android salga desde un punto determinado hasta que llegue a su destino, apoyado con un sistema que genere conocimientos en base a esta información, ofreciendo datos actualizados que incluyan las rutas más transitadas de la ciudad y que día y horas son las de mayor tráfico.

Esto permitirá poner al servicio del usuario una base de conocimiento que pueda consultar, ayudándolo a saber que ruta tomar si el día de hoy es viernes y debe volver del trabajo o al hogar con tan sólo enviar un mensaje al servidor con el lugar de origen y el lugar de destino. El servidor podrá contener la información acerca de viajes previos, entre estos dos lugares, y puede proveer al usuario de promedios de tiempo en base a esa ruta.

Con todo lo descrito anteriormente, vale resaltar como la mayor ventaja el hecho de trabajar con el ambiente de desarrollo que ofrece la plataforma Android. Es una herramienta totalmente amigable y fácil de usar, la cual cuenta con poco tiempo en el mercado y presenta un entorno bastante dado al desarrollo de interfaces de usuario amigables debido a que posee

interacción con la pantalla táctil del teléfono, además de las funcionalidades que presta el sistema operativo como tal.

Una de las características que lo diferencia de otras plataformas, es que Android toma todas las actividades relacionadas a una aplicación y las liga a un solo proceso. Como nuestra aplicación estará ligada a una serie de actividades o tareas independientes de forma simultánea, esta característica resulta altamente conveniente, debido a que no se abrirá un proceso por cada actividad, lo cual consumiría muchos recursos del procesador y crearía procesos innecesarios.

Otra característica importante es que Android utiliza Dalvik Virtual Machine. Esta máquina virtual fue diseñada para el uso de requerimientos de baja memoria, lo que la hace más pequeña y la diferencia de otras máquinas virtuales para móviles. Nuestra aplicación debe ser Lite, es decir, que consuma bajos recursos, debido a que será una aplicación residente siempre activa que disparará eventos en cualquier momento. Ante esto, aprovecharemos esta característica de la máquina virtual para el desarrollo de la aplicación.

Resumiendo, Android es una tecnología nueva y con un gran campo lleno de posibilidades para realizar aplicaciones que no se hayan desarrollado antes. Por ser una plataforma de còdigo abierto abre la posibilidad de que la aplicación pueda expandirse a medida que otros programadores tengan acceso al código y den su aporte a la misma. Igualmente, tendrá la posibilidad de crecer junto con la tecnología, a medida que la plataforma evolucione. Apoyando esto, se seleccionó esta plataforma debido a que la aplicación interactuará con un dispositivo GPS y un registro de lugares, lo cual es favorable ya que esta tecnología posee una interacción única con Google Maps (aplicación de Google), interacción que no poseen otros sistemas operativos para dispositivos móviles. Esto permite poner a la disponibilidad del usuario y del desarrollador una serie de mapas y eventos relacionados a ellos.

Es importante resaltar que la plataforma Android presta un emulador sumamente completo, que facilita al desarrollador la creación de sus aplicaciones. Muestra el típico

hardware y software, y provee la navegación a través del teléfono virtual con el teclado y mouse. De esta manera, se pueden correr varias aplicaciones o generar eventos al mismo tiempo en el emulador. Esto fue de suma importancia al momento de la selección de la plataforma para realizar la aplicación, debido a que el emulador cubre prácticamente todas las funcionalidades del teléfono, por lo cual se podrá probar la aplicación sin problema a medida que se vaya desarrollando.

Todo el proceso nos permitirá ver a ciencia cierta las ventajas de esta tecnología en cuanto al desarrollo libre de aplicaciones para dispositivos móviles, en comparación con otras plataformas similares. Para esto, se adquirirán los equipos necesarios para poder trabajar con esta tecnología en auge, contando así con todos los recursos para ofrecer una solución totalmente funcional, que cuente con originalidad e innovación, a este nuevo mundo de los sistemas Android.

Limitaciones y alcances

Limitaciones

- Contamos actualmente con la versión 1.5 Release3 del SDK de Android, utilizado para el desarrollo de aplicaciones de este tipo. Sin embargo como es una tecnología en pleno crecimiento, hay funcionalidades que aún no están implementadas.
- Aunque podría ser migrado a otros lenguajes, aun sólo el SDK de Android está disponible en el lenguaje JAVA de programación.
- Debido a lo nuevo de la tecnología, podría no haber mucha documentación disponible acerca de los recursos necesarios al momento de la programación.
- Los mapas que existen para Venezuela no son muy detallados, lo cual puede afectar la exactitud del GPS.
- El GPS siempre tendrá un rango de error en cuanto a las coordenadas, por lo cual puede haber un pequeño desfase en las distancias calculadas.
- Una lluvia fuerte y humedad extremadamente grande pueden afectar la precisión del GPS.

- Los eventos asociados a envío de correos electrónicos, se dispararán siempre y cuando el dispositivo móvil tenga configurada una cuenta de correo electrónico y se haya establecido la respectiva conexión.
- Los eventos asociados a envío de mensajes de texto, se dispararán siempre y cuando el dispositivo móvil posea señal de cobertura.
- El módulo de consulta de tráfico de la aplicación estará disponible siempre y cuando el servidor de mensajes de texto esté en funcionamiento.

Alcances

- Desarrollo de una aplicación bajo la plataforma Android, que cuenta con un sistema de tareas que permite configurar en el teléfono, radios de acción alrededor de coordenadas registradas a través del GPS del dispositivo móvil.
- El equipo con el que se trabajará será el HTC Dream G1, celular con las especificaciones necesarias para implantar la aplicación.
- Cómo la aplicación es desarrollada en la plataforma Android, sólo funcionará y será compatible con esta plataforma. El sistema se relaciona en todo momento con las funcionalidades que esta plataforma ofrece, logrando un cómodo uso de la tecnología elegida.
- La aplicación, cuenta con dos partes principales:
    - Sistema de tareas: Este sistema cuenta con una estructura de almacenamiento que contiene los datos acerca de los lugares de preferencia del usuario con sus determinadas coordenadas. Con estos lugares registrados, se tendrá la opción de crear una serie de tareas las cuales pueden comprender alarmas, recordatorios, envío de mensajes de texto por posicionamiento y envío de correo electrónico.
    - Servidor de mensajes de texto y consulta de tráfico: se pueden hacer consultas de tráfico con el dispositivo móvil. La información del tráfico está en una base de datos en un servidor de mensajes de texto, que estará actualizada en todo

momento con datos acerca de una serie de rutas mapeadas de la ciudad de Caracas.

- Se cuenta como documentación adicional, un estudio de comparación entre la plataforma Android y otras plataformas análogas al sistema operativo Android existentes en el mercado: *Limo, Symbian* y *OS X Mobile*, así como la aplicación desarrollada con el presente proyecto y aplicaciones de funcionalidades similares implementadas y desarrolladas. El estudio se basará principalmente en la búsqueda de ventajas y desventajas en cuanto a las funcionalidades encontradas, desarrollo, usabilidad, entre otras.

CAPÍTULO II
MARCO DE REFERENCIA / TEÓRICO

Tecnología GPS

Hoy en día existen una gran cantidad de tecnologías que están al alcance de cualquier usuario alrededor del mundo. Una muy importante y que está tomando mucha fuerza como tecnología afluente en la actualidad es el Sistema de Posicionamiento Global o GPS, como es más conocido. Este tan novedoso sistema, que tiene vigencia desde aproximadamente 1957, permite saber la posición de una persona, objeto o nave con una precisión casi exacta. Esto se da usando un GPS diferencial, el cual se presenta como un sistema que proporciona correcciones de los datos recibidos a nivel satelital, pudiendo llegar así a una mayor precisión en la posición que se quiera obtener. Este sistema funciona igualmente con ayuda satelital. Gracias a una red de veintisiete (27) satélites en órbita sobre el globo, se puede cubrir toda la superficie de la Tierra. El funcionamiento es más sencillo de lo que podría esperarse:

"...Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo tres satélites de la red, de los que recibe unas señales indicando la posición y el reloj de cada uno de ellos. Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el retraso de las señales; es decir, la distancia al satélite. Por "triangulación" calcula la posición en que éste se encuentra."

(http://es.wikipedia.org/wiki/DGPS#GPS_Diferencial)

También, esta tecnología cuenta con el apoyo de ciertas características técnicas las cuales son:

- Sistema de satélites que está conformado por aproximadamente veinticuatro (24) unidades, sincronizados para cubrir toda la superficie terrestre.

- Estaciones terrestres que se encargan de controlar las órbitas.
- Terminales receptores, los cuales son conocidos también con el nombre de Unidades GPS, que se encargan de indicar la posición en la que se encuentran.

Ubicación al alcance de todos

Adicionalmente, los tiempos han cambiado. No desde hace mucho tiempo la tecnología GPS estaba reservada para ciertos sectores sociales. Sin embargo, rápidamente se ha convertido en algo que está al alcance de cualquier persona. Entre estas nuevas tecnologías, resalta la conocida Google Maps, como una de las últimas invenciones de la compañía Google. Es un servidor de aplicaciones de mapas Web que ofrece imágenes satelitales del mundo entero, mapas desplazables y las rutas entre miles de millones de lugares alrededor del mundo. Por sobre todas las cosas, y es de lo que se hablaba anteriormente, es un servicio totalmente gratuito y al alcance de cualquier persona.

Aunque esta tecnología está disponible vía Web para todos los usuarios, también está disponible para dispositivos móviles. Con esto, cualquier persona con un teléfono celular compatible puede tener a su alcance información local, rutas y mapas de la región donde vive, donde trabaja o cualquier zona que esté frecuentando.

Tecnología Android

Como otra tecnología, a nivel de dispositivos móviles de bastante auge en los últimos tiempos, podemos mencionar la plataforma Android, la cual es otra útil invención desarrollada también por la compañía Google. Se trata de una plataforma de software y un sistema operativo para celulares y dispositivos móviles. Es una tecnología relativamente nueva (fue presentada en noviembre del año 2007) la cual permite, entre otras funcionalidades, escribir el código en lenguaje Java con librerías desarrolladas también por Google, las cuales están destinadas a controlar el dispositivo móvil.

Ofrece una serie de características que sólo dicen un poco de la casi infinitas funcionalidades que se le podría dar. Las características incluidas, expuestas en la página de código abierto de Android de Google, son las siguientes:

- Framework de aplicaciones (Con uso y reemplazo de componentes).
- Máquina virtual *Dalvik* (Optimizada para dispositivos móviles).
- Navegador integrado (Basado en el motor *Open Source WebKit*).
- Gráficos optimizados.
- *SQLite* para almacenamiento de datos estructurados. Esta característica permite crear y manejar base de datos locales de una forma fácil y rápida.
- Soporte para medios con formatos comunes de audio, video e imágenes planas (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- Telefonía GSM (Dependiente del hardware).
- Bluetooth, EDGE, 3G, y Wi-Fi (Dependiente del hardware).
- Cámara, GPS, brújula, y acelerómetro (Dependiente del hardware)
- Ambiente rico de desarrollo incluyendo un emulador de dispositivo, herramientas para depurar, perfiles de memoria y rendimiento, y un plugin para el IDE Eclipse.

(Google Android, http://code.google.com/intl/es-ES/android/what-is-android.html)

Esta tecnología presenta una arquitectura bastante definida, donde se pueden hallar las posibilidades de trabajo y programación con las cuales se puede contar:

1. Las aplicaciones, las cuales incluyen a un cliente de email, programa de SMS, calendario, mapas, navegador, contactos y otros.
2. Un Framework de aplicaciones, cuyos API's pueden ser accedidos completamente por los desarrolladores.
3. Librerías en C/C++, que son usadas por varios de los componentes del sistema Android. Estas capacidades son expuestas a los desarrolladores a través del Framework de aplicaciones.

4. El Runtime de Android, que incluye un set de librerías base que proveen la mayor parte de las funcionalidades disponibles en las librerías base del lenguaje de programación Java.
5. También cuenta con un Núcleo basado en Linux, del cual depende para los servicios base del sistema (seguridad, gestión de procesos y memoria, entre otros).

Figura 1. Arquitectura Android.

Fuente: http://code.google.com/intl/es-ES/android/what-is-android.html

Actualmente la tecnología Android es totalmente pública mediante el software libre de Apache y su licencia de código abierto. Esto permite que las posibilidades de trabajar con esta tecnología se expandan cada vez más, permitiendo así a miles de usuarios tener la oportunidad de desarrollar nuevas y mejoradas aplicaciones basadas en este software. Para esto se ha desarrollado el Android Software Development Kit (SDK), el cual en su versión Beta, provee de las herramientas y los API's necesarios para comenzar a desarrollar aplicaciones en la plataforma de Android usando el lenguaje de programación Java.

Esto ha permitido que un gran número de usuarios puedan desarrollar aplicaciones sobre esta plataforma, incluso logrando así, entre muchos ejemplos, la nueva versión del mencionado Google Maps específica para el T Mobile G1 Android, que logró sacar el máximo provecho de todo su hardware, como la pantalla táctil, el acelerómetro, el GPS y la aceleración de gráficos.

Android Software Development Kit (SDK)

EL SDK de la tan reciente tecnología Android es un Framework bastante completo con muchas herramientas que pueden usarse para la programación. El SDK está publicado bajo la licencia Apache 2.0.

De manera general, el SDK está conformado por tres directorios principales:

1. Documentación: esta documentación cuenta con los pasos mínimos que hay que conocer para poder programar, la instalación del mismo SDK, la anatomía de una aplicación y las herramientas de desarrollo.
2. Desarrollando Aplicaciones: en esta sección se explica claramente cómo realizar la implementación de una interfaz gráfica, cómo realizar el acceso a datos, entre otros recursos.
3. Caja de Herramientas: este directorio cuenta con la información necesaria acerca de la filosofía de diseño, API's (de Google y opcionales) y todo lo relacionado con la construcción de componentes. La caja de herramientas también cuenta con uno de los elementos más importantes: El emulador.

Figura 2. Escritorio del Emulador que ofrece la plataforma Android.

Fuente: Captura propia del ambiente de desarrollo utilizado

El emulador de esta plataforma, cuenta con una representación del dispositivo móvil bastante completa, dando acceso a las distintas herramientas del terminal: aplicaciones, vistas, contactos, explorador, mapas, devTool, y mucho más.

Debido a que es una tecnología en constante crecimiento, son muy frecuentas las constantes actualizaciones del SDK de Android. Actualmente se encuentra en su versión 1.5 Release 3, la cual fue lanzada en el pasado Julio de 2009, incluyendo nuevas funcionalidades: grabar y ver videos, subir videos a YouTube, subir imágenes a Picasa, nuevo teclado ligero, conexión automática con el Bluetooth, widgets y carpetas, animaciones entre pantallas, etc.

Empezando el desarrollo: Eclipse y Android

Actualmente el SDK tiene una perfecta integración con Eclipse, un muy usado entorno de desarrollo de código abierto. Esto permite también que la instalación de esta tecnología sea totalmente sencilla y muy rápida. También, es muy fácil la actualización en caso de presentarse, logrando una perfecta migración a versiones más recientes del SDK de Android.

El proyecto Android cuenta con una "Actividad" principal, que representará una pantalla dentro de la aplicación. Un mismo proyecto puede contar con todas las actividades que deseen. Asociado a una actividad, se crean también archivos XML, que contendrán lo que son las características de la interfaz gráfica de la pantalla. Para definir finalmente con todo lo que se cuenta en un proyecto Android, se puede listar todos los componentes:

- Carpeta Source: incluye los paquetes con los archivos JAVA con el código perteneciente a cada una de las actividades que conformarán la aplicación final.
- Carpeta Gen: aquí se encuentran todos los archivos JAVA que son generados automáticamente generados.
- Google API's: aquí se encuentran todas las librerías que ofrece Google para el desarrollo.
- Carpeta Res: contiene a su vez carpetas en las que podremos encontrar los distintos íconos gráficos y las plantillas asociadas a cada actividad que conforma la aplicación:

- Drawable: contiene todas las imágenes e íconos que podrían integrarse en la interfaz gráfica de la aplicación.
- Layout: contiene los archivos XML que serán editados para lograr la apariencia gráfica de cada una de las pantallas que conformarán la aplicación final. Estos archivos pueden ser editados en el mismo Eclipse tanto a nivel de código como a nivel gráfico.
- Values: aquí pueden ser agregados en archivos XML todos aquellos valores que sean considerados constantes en el proyecto en cuestión.

- Android Manifest: este archivo XML contiene todos los datos importantes de la aplicación: los paquetes, la versión del código, el nombre de la aplicación, las actividades que las conforman, los permisos y restricciones e incluso registro de variables y demás.

Figura 3. Proyecto nuevo del tipo Android en Eclipse

Fuente: Captura propia del ambiente de desarrollo utilizado

Por otra parte, para lo que es el manejo de base de datos en Android se tiene implementado el uso de bases de datos de tipo SQLite, que se presenta como un manejador de tipo relacional que es enlazado a la aplicación Android a través de la parte integral del mismo mediante llamadas a rutinas y subrutinas. Un análisis de las bases de datos SQLite completo puede ser consultado en el APÉNDICE D, adjunto a este documento.

Envío y recepción de mensajes de texto

Entre los objetivos de este proyecto se contempla la instalación de un servidor de mensajes de texto. En los tiempos que se viven actualmente hay muchas maneras de lograr esta instalación. Las distintas compañías que ofrecen servicios de telefonía celular tienen entre sus ventas equipos llamados "Modem de banda ancha" que ofrecen muchas funcionalidades que los usuarios podrían disfrutar.

Este tipo de módems cuentan con muchas funcionalidades, entre ellas el Servicio de SMS. Contando con una cuenta, y un chip de la compañía de telefonía celular y por supuesto el saldo suficiente, esta tecnología permite el envío y recepción de mensajes de texto fácilmente desde el computador. Incluyen programas de instalación rápida que acompañan estos módems, que permite que estas acciones sean lo más fácil posibles de gestionar y el usuario pueda obtener el máximo de este tipo de tecnologías. Sin embargo, también en la codificación esto es posible. Existen actualmente muchas librerías, y la plataforma Android la contiene en sus API's, que permiten acceder a los mensajes que salen y llegan a un dispositivo móvil. Esto todo se integra fácilmente una vez que se ha codificado correctamente un servidor que se encargue de gestionar la salida y llegada de mensajes de un celular. Eso es perfectamente integrable con algún programa o aplicación que se esté desarrollado, para integrar esta característica tan importante en el funcionamiento del dispositivo móvil en el sistema que algún usuario se disponga a desarrollar.

CAPÍTULO III
MARCO METODOLÓGICO

Para el desarrollo de la aplicación, se siguieron todos los pasos del ciclo de vida del desarrollo de la misma. Se comenzó el camino con el levantamiento de requerimientos, el análisis y el diseño global. Luego de estas tres etapas se estimó seguir una metodología lineal que permitiera el confiable desarrollo de la solución. Sin embargo, las tres etapas mencionadas anteriormente son etapas que deben ser evaluadas constantemente con el objetivo de reducir al máximo los riesgos para lograr dar con el resultado más óptimo.

Es por esto que se eligió el Modelo Cascada con reducción de riesgos como metodología para el desarrollo del presente proyecto. Este modelo ayuda cuando en el camino se encuentran inconvenientes en cuanto a la definición de requerimientos antes de comenzar con el diseño de la arquitectura. El modelo de Cascada con Reducción de Riesgos, realiza iteraciones entre las fases de análisis y diseño con el fin de controlar que se haya hecho el adecuado levantamiento de información y el diseño corresponda con éste. Para ello se desarrollaron prototipos de interfaz de usuario que fueron evaluados para identificar los requerimientos exactos del sistema. Con estas modificaciones del modelo cascada clásico se logró mayor flexibilidad en el desarrollo de la aplicación:

"Para evitar los problemas que se presentan por la rigidez del modelo en cascada, se puede establecer una cascada con una espiral de reducción de riesgos para el análisis de requerimientos o el diseño de la arquitectura"

(Steve McConnell, 1997. *Desarrollo y gestión de Proyectos Informáticos*. McGraw-Hill)

En este modelo escogido, Modelo Cascada con Reducción de riesgos, se contaron con las siguientes etapas en el ciclo de vida de la aplicación:

- Análisis de Requerimientos: esta etapa se refiere a lo que sería la recopilación de los requisitos. Se centra totalmente e intensifica especialmente en el software o solución a

buscar. Se debe comprender a cabalidad el ámbito de la información del problema, así como las necesidades del cliente o usuario final.

- Diseño Global: con los requerimientos ya conocidos, se procede a la realización de un diseño preliminar de la solución al problema planteado. Se deben traducir los requisitos en una representación del software con la calidad requerida, para luego seguir con la codificación.
- Diseño detallado: se completa el diseño preliminar contemplado en el diseño global. Se debe enfocar en la estructura de datos, arquitectura, detalles procedimentales y la caracterización de la interfaz a implantar.
- Codificación y depuración: una vez que se cuenta con el diseño, el mismo debe traducirse en una forma legible para el computador. Si el diseño se realiza de una manera detallada, la codificación y depuración puede realizarse mecánicamente.
- Prueba del sistema: una vez que se ha generado el código, se puede proceder a realizar las pruebas del software. Se deben centrar en la lógica interna y en las funciones externas, asegurándose así que la entrada definida produce los resultados que realmente se requieren.
- Mantenimiento: el software sufrirá cambios después de que se entrega al cliente. Estos cambios son ocasionados por el encuentro con nuevos errores, de esto se trata el mantenimiento de la solución que fue llevada al software.

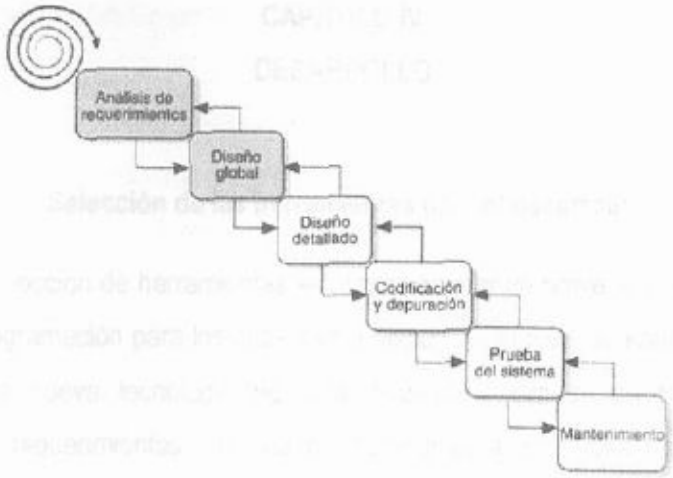

Figura 4. Modelo Cascada con Reducción de riesgos.

Fuente: Steve McConnell, 1997. *Desarrollo y gestión de Proyectos Informáticos*. McGraw-Hill

Cómo se puede observar en la Figura 5, el Modelo Cascada con reducción de riesgos, la reducción de riesgos no sólo está limitada para las dos primeras etapas, sino que puede llevarse a cabo en referencia en cualquier otro riesgo del proyecto, a lo largo de todo el ciclo de vida, lo cual permite estar en constante actualización en cuanto a los requerimientos del proyecto en cuestión, para así lograr una solución lo más acertada posible para el cliente y usuario final, mantenimiento la linealidad del modelo de cascada pura, logrando ofrecer un modelo de planificación fácil y sencillo conformado con fases conocidas para los desarrolladores y de fácil entendimiento para los usuarios.

CAPÍTULO IV
DESARROLLO

Selección de las herramientas para el desarrollo

Para la selección de herramientas se procedió a ser un breve estudio de los distintos ambientes de programación para los cuales está disponible el SDK de Android. Se descubrió que por ser una nueva tecnología no está disponible para su desarrollo en muchas plataformas. Los requerimientos mínimos de sistema para la exitosa programación en el ambiente Android son los siguientes:

- Windows XP (32-Bit) o Vista (32 o 64-bit)
- Mac Os X 10.4.8 o posteriores versiones (sólo x86)
- Linux (probado en Linux Ubuntu Dapper Drake)

Estas opciones fueron evaluadas y finalmente la decisión se inclinó hacia el lado de la opción de Windows XP, puesto que representa el ambiente donde más experiencia se ha tenido y se contaba también con los equipos con los requerimientos necesarios, disponibles y listos para ser utilizados. Una vez seleccionados los requerimientos y el software a utilizar para el desarrollo de aplicaciones Android, se procedió a hacer una evaluación de lo que serían los ambientes de programación soportados:

- Eclipse 3.3 (Europa), 3.4 (Ganymede)
    - Paquetes Eclipse IDE recomendados: Eclipse IDE para desarrolladores Java EE, Eclipse IDE para desarrolladores Java, Eclipse para desarrolladores RCP/Plug-in.
    - Eclipse JDT Plugin (incluido en la mayoría de los paquetes Eclipse IDE).
    - No es soportado en el paquete Eclipse Classic IDE
- JDK 5 o JDK 6 (JRE sólo no es suficiente)
- Android Development Tools Plugin (opcional)
- Otros ambientes de desarrollo o IDE's

- Apache Ant 1.6.5 o posterior para Linux y Mac, 1.7 o posterior para Windows.

Finalmente se seleccionó Eclipse 3.3 (Europa) como ambiente de desarrollo final para la programación de la aplicación Android que representaría la solución del problema planteado. De nuevo, fueron evaluadas las ventajas de este ambiente, así como también influyó la experiencia previa ganada en el pasado en relación a Eclipse 3.3 (Europa). Con las opciones de software ya elegidas, se consultó lo que sería la instalación del SDK en el ambiente Eclipse 3.3 (Europa), la cual resultó ser bastante sencilla y rápida. Para mayor información sobre la instalación del SDK en este ambiente de desarrollo, consultar el APÉNDICE A, en donde también se podrá consultar las vías de actualización del SDK el cual como se dijo está en constante cambio con nuevas versiones.

Con la plataforma Android seleccionada para la codificación, el tipo de sistema operativo en el que se trabajará y el ambiente de desarrollo determinado, ya se tenía un completo contexto de programación para comenzar con el desarrollo de la aplicación. Sin embargo, se trata de una solución que estaría enfocada para dispositivos móviles, por lo cual fue necesario encontrar un equipo que sirviera para futuras pruebas de la aplicación e incluso para la instalación de versiones previas de la misma. Después de evaluar las opciones, se tuvo la oportunidad de adquirir el modelo HTC Dream G1, el cual representaba el primer móvil que adoptaba las especificaciones necesarias del sistema operativo elegido, el opensource Android.

Figura 5. Dispositivo HTC Dream G1 Android de Google

Fuente: http://e-global.es/b2b-blog/2008/09/18/htc-dream-g1-de-google-con-android-mas-informacion/

No sólo representó la mejor opción de equipo para la programación, sino también contaba con todas las especificaciones perfectas como dispositivo móvil en cuanto al localizador GPS, aplicaciones Google, y demás requerimientos contemplados en la aplicación a desarrollar.

Característica	Tipo contemplada en el dispositivo
Procesador	528Mhz Qualcomm 7201
Memoria	64MB RAM Interna / 128 ROM Interno
Tarjeta de expansión	MicroSD de 1 GB
Conectividad	3G y Quad Band (850, 900, 1800, 1900) y EDGE en zona de cobertura
Cobertura	Wifi, WiMAX
Bluetooth	2.0 Estéreo
SMS, MMS	Soporte para mensajes cortos SMS y mensajes de imagen MMS
Teclado	QWERTY de cinco filas y rueda TrackBall
Cámara de fotos	3.1 mega píxeles, sin flash, a 2048 x 1536 y botón dedicado.
Video playback	H.264, streaming, 3GPP, MPEG4 y Codec 3GP / Reproductor dedicado de Youtube
Ringtones	MP3, AAC, AAC+, WMA / Altavoces en mono.
Reproducción de audio	Formatos soportados: MP3, AAC, AAC+, WMA, MPEG4, WAV, MIDI, REAL AUDIO, OGG
Formatos de gráficos /imagen	MPEG4, H.264, MP3, AAC, AMR, JPG, PNG y GIF
Gráficos optimizados	Para 2D y #D basados en software OpenGL
Correo electrónico	Soporte para POP, IMAP y SMTP.
Mensajería	AIM, MSN, YAHOO y GTALK
Pantalla	3,17 pulgadas con HVGA (480 x 320).
Sincronización	Google Calendar, Android Marketplace, Goggle Streetview y Google Maps
Navegador	HTTP, WAP Push, xHTML
Características adicionales	Brújula, Picture Caller ID, Sistema de posicionamiento satelital GPS, Acelerómetro

Tabla 1. Características conocidas del dispositivo móvil HTC Dream G1 Android de Google

Fuente: http://e-global.es/b2b-blog/2008/09/18/htc-dream-g1-de-google-con-android-mas-informacion/

Por otro lado también se necesitó la selección de un servidor de mensajes de texto. Se evaluaron muchas opciones, entre ellas la compra de un modem de la compañía Movistar. Sin embargo, en pruebas previas para evaluar la tecnología, se tuvo acceso a uno y no se logró la conexión con el mismo por problema de configuración y conexión con los equipos utilizados para el desarrollo de la aplicación.

Finalmente, después de investigar al respecto, se encontró que cualquier equipo o dispositivo móvil que tuviera la conexión correcta al computador que se usaría, podría funcionar correctamente como servidor de mensajes de texto para la realización de pruebas de la aplicación. Ante esto, se encontró un equipo de desuso que se pudo implementar para la realización de pruebas durante el desarrollo del presente proyecto. El equipo encontrado pertenece a la familia Motorola en su modelo C650. No es un equipo muy actual, pero cumplió con los requerimientos mínimos que se necesitaban para usarse en el desarrollo de la aplicación.

Especificaciones del equipo Motorola C650	
Dimensiones	102,5 x 44 x 19,6 mm
Peso	91,5 g.
Año de lanzamiento	2004
Sistema interno	Tribanda
Tarjeta SIM	Plug-in
Memoria RAM	1,8 MB
Dimensiones de la pantalla	128 x 128 pixeles
Número de colores	65.000 colores
Batería	Li-ion
Autonomía en llamada	6 horas y 20 minutos
Autonomía en stand-by	215 horas
Otras especificaciones	Pantalla a color, Java, Vibracall, EFR (Enhanced Full Rate), GPRS, Infrarrojo, WAP, Modem, Datos/Fax, EMS, MMS, T9

Tabla 2. Especificaciones del equipo móvil Motorola C650

Fuente: http://www.sangrefria.com.es/2006/02/05/motorola-c650/

Una vez seleccionado el equipo que representaría el servidor, se comenzó la evaluación de cómo sería la implantación de dicho servidor a nivel de código en el proyecto. Después de evaluar una serie de alternativas, finalmente, se tomo la solución JSms Versión 1.0, el cual es un servidor de SMS que fue desarrollado para la gestión de mensajes de texto, por la Website Baker, un sistema de manejo de datos basado en PHP encargado de lograr que los usuarios produzcan sitios web de una manera fácil y rápida. Esta aplicación es capaz de establecer una comunicación a través de un puerto serie de la computadora y manejar así uno o varios módems conectados al computador. Permite diferenciar cuando se trata de un módem de entrada y uno de respuesta. Los módems de entrada se encargan de recibir los mensajes de texto y los de respuesta de responderlos. También está permitido el uso de un mismo módem para ambas tareas. Este servidor está disponible actualmente para su descarga gratuita desde un sitio en línea de la Website Baker.

INTEGRACIÓN DE LA APLICACIÓN CON GOOGLE MAPS

Análisis de requerimientos

Después de identificar el problema planteado, se llegó a listar una serie de requerimientos completa, que giran en base a la inquietud de que los usuarios puedan configurar tareas en base a su actual ubicación en el mundo. Se requiere entonces que los usuarios finales de la solución puedan ser capaces de consultar su ubicación, valiéndose del dispositivo GPS de su teléfono móvil. Es por eso que el usuario debería poder ver fácilmente en la aplicación su ubicación en el mundo, de la forma más gráfica, fácil y entendible posible. Igualmente, que pueda ser capaz de moverse en el espacio y aún así darse cuenta de que su ubicación ha cambiado y que pueda a través de la aplicación darse cuenta que esto ha pasado. Finalmente el usuario debería ser capaz de poder buscar alguna ubicación de su preferencia dentro de los mapas disponibles, mediante alguna palabra clave, o alguna otra información.

Diseño global

Tomando en cuenta el análisis de los requerimientos, se deben especificar claramente las entradas y salidas de lo que sería la pantalla principal de la aplicación: la consulta de la locación o ubicación actual del usuario en el mundo. El usuario debería ingresar mediante alguna opción una solicitud de locación, y que tenga como salida su ubicación exacta. No hay manera más clara de darle a conocer esto al usuario, que mediante mapas. Por lo cual el diseño global debería centrarse en una serie de mapas en donde el usuario pueda consultar su ubicación actual, de una manera clara, fácil y sin demasiadas transiciones, teniendo la oportunidad de no sólo consultar esa ubicación actual sino también cualquiera que el usuario pueda ingresar mediante palabras clave.

Diseño detallado

Debido a que se trata de un programa para dispositivos móviles, el usuario verá la aplicación en distintas pantallas de la misma. El diseño de esta parte de la aplicación, constaría de una pantalla inicial de la aplicación que mostrará una serie de mapas. Consultando con aplicaciones en línea, relacionadas con el posicionamiento GPS, como Google Maps y Google Earth, se encontraron varias opciones que sería bueno implementar para mayor disfrute del usuario de la aplicación, como serían controles de Zoom, diferentes tipos de vistas y medios gráficos para mostrar la ubicación solicitada. Ante esto, estas opciones fueron consideradas en el diseño.

Tomando en cuenta lo anteriormente analizado, mediante un menú, el usuario podrá acceder a las principales opciones del mismo que serían las siguientes: mostrar ubicación actual del dispositivo, mostrar la vista mapa, mostrar la vista satelital, buscar ubicación en el mapa y salir de la aplicación. La opción de ir a la ubicación actual le mostrará al usuario en el mapa, de una manera visible, cual es su ubicación actual. Las opciones vista mapa o vista satelital le darán la oportunidad al usuario de ver los mapas según sea su necesidad.

Finalmente la opción de buscar ubicación le dará la oportunidad al usuario de mediante palabras claves encontrar ciertos lugares en los mapas disponibles.

Codificación y depuración

Al momento de la codificación se procedió a investigar sobre lo que sería la elaboración de una aplicación basada en Mapas en Android. Debido a que Android es una aplicación creada por Google, tiene las herramientas necesarias para la perfecta integración de la misma con Google Maps, para la proyección de mapas dentro de las aplicaciones que se desarrollen. Se cuenta con las librerías ya integradas para hacer una completa aplicación basada en mapas. Entre esas, está la librería/clase llamada "MapView", que se encuentra en la librería externa y la cual es bastante útil. Provee de herramientas de construcción y descarga de mapas así como una variedad de opciones y controles adicionales. También permite hacer solicitudes y manipular la data de Google Maps a través de métodos de clase.

Se debe destacar, que para la correcta visualización de los mapas dentro de una aplicación, se debe contar con un API Key, la cual es una clave de seguridad que permite el manejo de los mapas dentro de una aplicación, y está asociada a cada versión del ambiente de desarrollo con el que se esté trabajando. Esto sirve para registrar el servicio de Google Maps, para que así la "MapView" sea capaz de obtener datos de Google Maps. Esta API Key es de fácil obtención con la huella digital de entorno de desarrollo utilizado. Para consultar los pasos a seguir para conseguir esta API Key, se puede consultar el APÉNDICE B.

Finalmente, una vez configurado el "MapView" de la pantalla principal, y con la API Key ya especificada, al correr la aplicación en el emulador se pudieron visualizar por primera vez los mapas, resultado de la integración final de la aplicación en desarrollo con Google Maps. Mediante la consulta de las fuentes necesarias, se pudo agregar lo que se necesitaba en cuanto a controles de los mapas: Zoom, Vistas y Ubicación actual del dispositivo móvil.

En esta parte de la aplicación se tuvo una constante reducción de riesgos, puesto que por la constante actualización del software y de las nuevas versiones de Android, se tuvo que

actualizar el código y añadir nuevas funciones y opciones según fueran las nuevas configuraciones añadidas. Finalmente, todas estas funcionalidades fueron integradas en la clase principal "Aplicación.java", la cual sería la actividad que representaría al que llamaremos Módulo Principal, el cual sería el centro de la aplicación. También se agregó una última funcionalidad: una nueva clase llamaba "Buscar.java", con la cual se pudo codificar la necesidad del usuario de buscar, mediante alguna palabra clave, alguna ubicación dentro de los mapas disponibles. Esta ubicación encontrada, es mostrada en el Mapa con una etiqueta visible para el usuario, centrando la pantalla en ese nuevo punto. Las capturas de las pantallas de la aplicación, así como sus casos de uso pueden ser consultados en el APÉNDICE H, adjunto al presente documento.

MÓDULO DE REGISTRO DE COORDENADAS

Análisis de requerimientos

Ya habiendo codificado lo que sería la pantalla principal de visualización de mapas dentro de la aplicación a desarrollar, también se encontró la necesidad de que los usuarios pudieran tener un registro de sus lugares de preferencia. Ya sea el hogar, el trabajo o la universidad, el usuario debería ser capaz de registrar cualquier lugar en donde se encuentra, en el momento que éste lo requiera.

Igualmente, también el usuario debería ser capaz de guardar las coordenadas de cualquier lugar que pueda ser visualizado en los mapas. Estas coordenadas registradas deberían ser asociadas a un lugar y una serie de datos que el usuario pueda especificar. El usuario debería ser capaz de poder actualizar estos datos cuando le sea necesario, poder ingresar nuevos lugares cuando requiera o eliminar lugares que ya no frecuente en el momento que le sea requerido.

Diseño Global

En el diseño global, se definen las entradas y salidas que debe tener el especificado Módulo de Registro de Coordenadas. El usuario debería poder hacer una solicitud mediante una opción de menú que le permita guardar la ubicación actual que tiene el dispositivo móvil. Captando las coordenadas del lugar en cuestión, debería permitir que el usuario ingrese los datos asociados a este lugar, para luego ser almacenados.

Por otra parte, también el usuario debería poder moverse por los mapas disponibles, y hacer una solicitud en la que se especifique que se quieren guardar las coordenadas de un lugar que no es la ubicación actual del dispositivo móvil, pero que se requieren almacenar como un nuevo registro en la base de datos. Viendo lo anterior, el usuario deberá también mediante una opción de menú, poder consultar los lugares registrados en el dispositivo, y ver los datos asociados, coordenadas y demás, y poder modificar cualquier de estos datos.

Diseño detallado

Basándose en el diseño global, es bueno determinar con cuales procedimientos se contarán para poder cumplir con las salidas que el usuario está esperando de la aplicación. La aplicación presentará una segunda opción que le permita guardar las coordenadas de su ubicación actual y asociarlas a un nuevo lugar dentro de la aplicación. Igualmente, el usuario podrá guardar cualquier ubicación que pueda visualizar en el mapa, tomar las coordenadas y almacenarlas. En este último caso, el diseño detallado contempla que la vista de mapas tenga un centro visible que permita guardar esas coordenadas una vez que el usuario lo haya movido hasta la ubicación de su preferencia. La configuración de este "centro" debería investigarse o en última instancia colocar una imagen fija en el centro de la vista de mapas.

Igualmente, para el correcto registro de coordenadas debería instalarse una base de datos que lleve todos los datos de los lugares que el usuario desee almacenar. Los datos que deberían ser almacenados en la base de datos, asociados a un lugar determinado son:

Nombre del campo	Tipo de dato
Id	Integer
Latitud	Double
Longitud	Double
Ciudad	String
País	String

Tabla 3. Campos de la tabla "Lugar" de la base de datos asociada a la aplicación a desarrollar

Fuente: Elaboración propia

Una vez diseñada la estructura de datos con la que se almacenaría la ubicación en la base de datos, es necesario el diseño de una arquitectura que permita el manejo de esta última. Para esto se contempla en el diseño una clase dedicada al manejo de las bases de datos de una manera genérica, la cual crearía, o abriría, en el momento de iniciar la aplicación la base de datos con las tablas que sean especificadas. También se debería definir una clase que se encargue de las distintas operaciones de lectura y escritura de la base de datos relacionadas con la tabla Lugar.

Finalmente, el diseño se resume en la pantalla principal en donde el usuario haría las solicitudes de registro y luego una segunda pantalla donde el mismo podrá ingresar los datos asociados al nuevo lugar a ingresar en la base de datos. La pantalla principal, añadiría a sus ya adquiridas funcionalidades la capacidad de poder tomar las coordenadas del centro cuando el usuario lo indique, la capacidad de poder almacenar las coordenadas de la ubicación actual del dispositivo móvil, y la posibilidad de consultar la lista de lugar que han sido almacenados en la base de datos.

Las opciones de menú actualizadas de la pantalla principal de la aplicación con el diseño detallado del Módulo de Registro de Coordenadas serían las siguientes: ir o guardar la ubicación actual del dispositivo, guardar las coordenadas del centro del mapa, mostrar la vista satelital o de mapa, buscar ubicación, consultar lugares y salir de la aplicación.

Las pantallas pertenecientes al Módulo de Registro de Coordenadas, acordando el actual diseño detallado, serán las siguientes:

- Pantalla de Registro: esta pantalla mostraría las coordenadas (latitud y longitud) capturadas del mapa que consultó anteriormente el usuario, y permitiría al usuario ingresar la Ciudad y el País, asociado al nuevo lugar que almacena. Esta pantalla tendrá dos funciones. El usuario podrá, según su solicitud, ingresar un lugar mediante esta pantalla o modificar los datos ingresados de un lugar ya existente.
- Pantalla de Confirmación: En esta pantalla el usuario podrá visualizar los datos que ingresó en su solicitud para ingresar un nuevo lugar, y confirmará si son correctos o no.
- Pantalla de Consulta: El usuario podrá visualizar una lista completa de todos los lugares almacenados en la Base de datos.

Para consultar con más facilidad como se realiza el flujo de información en el presente Módulo de Registro de Coordenadas, se recomienda la lectura del APÉNDICE C. Igualmente también en cada pantalla del Módulo de Registro de Coordenadas deben existir nuevas opciones de menú. Para la pantalla de registro se tiene la opción de guardar un nuevo lugar, consultar o editar alguno existente y salir de la aplicación. En la pantalla de modificar se tendrán las mismas opciones que en la de registro, con opciones adicionales para mostrar en el mapa el lugar almacenado o eliminarlo de la base de datos. En cuanto a la pantalla de confirmación y la de consulta, se puede guardar un nuevo lugar o salir de la aplicación. La pantalla de consulta permitirá agregar un nuevo lugar.

Codificación y depuración

Para el proceso de codificación y depuración del Módulo de Registro de Coordenadas se crearon tres nuevas clases en lenguaje de programación Java, que representarían las actividades de las tres nuevas pantallas que serían integradas en la aplicación:

- PantallaP.java: clase que manejará la pantalla de registro en donde el usuario podrá ingresar un lugar nuevo o modificar alguno ya existente en el sistema. La misma tiene asociada el archivo XML "pantallap.xml", en las que se especificaron las características que tendría la interfaz gráfica de esta primera pantalla.
- Pantalla2P.java: esta clase manejará la pantalla de confirmación, en donde el usuario verificaría los datos antes de ser almacenados en la base de datos.
- Pantalla3P.java: esta clase manejará la pantalla de consulta de los lugares que ya han sigo almacenados en la base de datos. Se podrá visualizar en esta pantalla una lista con todos los lugares que han sido almacenados. El usuario puede hacer click en cualquier de su preferencia, y así podrá actualizar algún dato asociado al lugar, con excepción de la latitud y la longitud los cuales no variarán de ninguna forma.

Las capturas de las pantallas de la aplicación, así como sus casos de uso pueden ser consultados en el APÉNDICE H, adjunto al presente documento. Una vez definidas estas clases se debió definir también aquellas clases que se encargarían de la interacción directa con la base de Datos. Para el manejo de base de datos, se utilizó la librería SQLiteDatabase de Android, que contiene todas las funcionalidades necesarias para el trabajo. Para una mayor documentación sobre el funcionamiento de las bases de datos en esta plataforma y en el presenta sistema, consultar el APÉNDICE D. Finalmente se crearon las tres primeras clases que estarían en el paquete de clases enfocadas para el manejo de datos dentro de la aplicación:

- DaoGenerica.java: esta clase contiene la apertura o creación de la base de datos cada vez que se corra la aplicación, las definiciones de las distintas tablas que contendrán los datos a crear durante el uso de la aplicación y el nombre de la base de datos a crear, para su posterior, si así se requiere, exportación.
- LugarDao.java: esta clase contiene todas las operaciones de lectura y escritura que se harán sobre la tabla "Lugar" de la base de Datos. Esta clase heredará funciones de la clase "DaoGenerica.java", definida anteriormente.

- Lugar.java: esta clase representará el objeto "Lugar" que se podrá manipular a lo largo de toda la aplicación.

Para consultar datos acerca de las clases codificadas y sus métodos, se recomienda la consulta del APÉNDICE F.

MÓDULO DE GESTIÓN DE TAREAS
MÓDULO DE GESTIÓN DE ALARMAS SIMPLES

Análisis de Requerimientos

Entre las primeras inquietudes del problema encontrado, móvil para la realización del presente proyecto, se encontró la necesidad de asociar de alguna manera una serie de alarmas a lo que sería la ubicación del dueño del dispositivo móvil en el mundo. Con la capacidad de la aplicación de guardar ciertos lugares de preferencia del usuario, se encontró la posibilidad de asociar esto a un sistema de tareas completo que permitiera darle nuevos usos al usuario del GPS, pudiendo así recordarle algo a cierta distancia de cierto lugar.

En el análisis de requerimientos se contempla entonces la creación de un Módulo de Gestión Alarmas Simples, que se encargue de asociar un mensaje de alerta a algún lugar que haya sido almacenado por el Módulo de Registro de Coordenadas y que sea mostrado cuando el dispositivo móvil del usuario entre en un radio de acción que éste último haya especificado con anterioridad. Las alarmas simples deben ser totalmente fáciles de configurar, siendo lo más amigable para el usuario y tratando de que éste pueda enterarse de lo que quiere, donde lo quiera y cuando lo quiera.

Diseño global

El usuario debe poder ingresar todos los datos asociados a una alarma, y registrarlos asociándolo a un lugar que ya exista en el Módulo de Registro de Coordenadas. Igualmente

debe ser capaz de poder consultar las alarmas cuando él lo requiera, y por consiguiente modificarlas si así lo desea. El registro de alarmas simples debe estar totalmente actualizado en todo momento en vista a las posibles modificaciones que pueda sufrir. Las salidas esperadas por el usuario deben ser la visualización de mensajes de alerta en el dispositivo móvil perfectamente visibles para el usuario que lo hagan recordar el mensaje que éste especificó, cuando pase el radio de acción del lugar que definió en el registro.

Diseño detallado

El Módulo de Gestión de Alarmas Simples constaría de una nueva pantalla en el cual el usuario deberá ingresar todos los datos asociados de la alarma Simple, los cuales serán luego almacenados directamente en la base de datos. En la documentación previa sobre la codificación en la plataforma Android, se encontró una definición bastante útil para la creación de estas alarmas: las alertas de proximidad. Después de analizar los requerimientos para la creación de este tipo de alertas, se definió la estructura del objeto que se manejaría a lo largo de todo el proyecto, y cuya estructura haría presencia en la base de datos del mismo:

Nombre del campo	Tipo de dato
Id	Integer
Asunto	String
Lugar Asociado	String
Radio de acción	Integer
Tipo de Alerta	String
Tono	String
Expiración	String
Error	String

Tabla 4. Campos de la tabla "Alarma Normal" de la base de datos asociada a la aplicación a desarrollar

Fuente: Elaboración propia

El asunto de una Alarma Simple, haría referencia al mensaje que el usuario quiere mostrar con la alarma configurada. El lugar asociado es el lugar en base al cual la alarma se disparará. ¿Cuándo se disparará? Cuando entre en el radio de acción que será también especificado. También se contemplaron las configuraciones de varios tipos de alerta en varios modelos de dispositivos móviles. Por lo tanto, se llegó a un diseño final en el cual el usuario pueda decidir si su alarma tendrá asociado un tono, si quiere que vibre al dispararse la alarma o si quiere que suene un tono y vibre a la vez. En cuanto a la expiración, se decidió que el usuario tuviera tres opciones posibles para definir la expiración de una alarma simple: un (1) día, siete (7) días, o que la alarma nunca expire y se mantenga en vigencia. Finalmente un campo de error, como control para el manejo de errores en el disparo de alarmas.

El usuario podrá configurar las alarmas que desee, podrá modificarlas cuando quiera, eliminarlas o consultarlas. Contemplando esto, el Módulo de Gestión de Alarmas Simples estará definido en dos pantallas:

- Pantalla de registro: aquí el usuario deberá ingresar los datos asociados de la alarma simple que quiere configurar. También esta pantalla se habilitará para la actualización de alarmas existentes. En su versión de ingresar el usuario podrá desde un menú consultar o editar las alarmas, volver a los mapas o salir de la aplicación. La pantalla de modificar además ofrecerá la opción para eliminar la alarma que se está modificando.
- Pantalla de consulta: en esta pantalla el usuario debe ser capaz de poder consultar las alarmas simples registradas. Como posteriormente al diseño principal serán añadidas otros tipos de alarmas, esta pantalla deberá hacer un filtro según sea la tarea. De esta pantalla la aplicación saltaría de nuevo a la pantalla de registro en su versión para modificar en donde el usuario pueda editar todos los valores de la alarma configurada. Las opciones de menú de esta pantalla permiten crear una nueva tarea, volver al mapa o salir de la aplicación.

Este módulo se relacionará directamente con el Módulo Principal. Del Módulo Principal, el usuario deberá poder hacer solicitudes a la aplicación para Crear nuevas tareas,

entre ellas la creación de nuevas alarmas simples y también deberá ser capaz de consultar la tarea que desee, especificando cual es las tareas disponibles en la aplicación quiere consultar. De nuevo, para una documentación más abierta acerca del flujo de información en el presente módulo, se recomienda la consulta del APÉNDICE C.

Después de esto, ciertas partes del diseño del Módulo Principal sufrieron modificaciones, sólo en cuanto a las opciones de menú se refiere, las cuales hasta los momentos serían las siguientes: ir o guardar la ubicación actual del dispositivo, guardar las coordenadas del centro, mostrar la vista satelital o de mapa, consultar lugares, buscar ubicación, crear o consultar tarea y salir de la aplicación. Como en el diseño general de todo el proyecto se contempla la creación de más Módulos de Gestión de Tareas, las opciones de "Crear tarea..." y "Consultar tarea..." serán opciones generales, en la cuales el usuario después debería especificar qué tipo de tarea quiere crear o consultar.

Codificación y depuración

Basándose en el diseño detallado, el Módulo de Gestión de Alarmas Simples está representado por dos pantallas principales, traducidos a clases en el lenguaje Java:

- PantallaAlarm1.java: esta clase representa la Pantalla de Registro del módulo. El usuario en esta pantalla podrá ingresar u modificar alguna alarma simple. Su interfaz gráfica está asociada a un archivo XML, llamado "pantallaalarm1.xml".
- PantallaConsultAlarm.java: esta clase representará la pantalla de consulta del Módulo de Gestión de Alarmas Simples. La codificación de esta clase fue hecha para que muestre la tarea que se haya decidido consultar. En el actual caso, el usuario visualizaría una lista con todas las Alarmas Simples almacenadas en la base de Datos, pudiendo observar el asunto de la Alarma y el lugar al que está asociada.

Las capturas de las pantallas de la aplicación, así como sus casos de uso pueden ser consultados en el APÉNDICE H, adjunto al presente documento. Una vez creadas las

pantallas y adelantado el proceso de codificación, se procedió a la creación de las clases que manipularían todas las operaciones de lectura y escritura de la información relacionada con las Alarmas Simples. En primera instancia se creó la clase que definiría el objeto "Alarma Simple", el cual sería manipulado a lo largo de toda la aplicación. La clase fue llamada "AlarmaNormal.java" y posee un procedimiento llamado "RegistrarAlarmasNormales", el cual ayudaría en lo que sería el registro de las alarmas configuradas a lo largo de toda la aplicación.

En cuanto a las clases que manipularían este objeto como tal, fueron creadas tres en total:

- AlarmaNormalDao.java: esta clase tendría todas los procedimientos y funciones que se necesitan para llevar a cabo operaciones de escritura y lectura de la base de datos.
- ProximityIntentReciver.java: esta clase contiene las acciones que se dispararán una vez que la Alarma Simple sea activada.
- Alerts.java: esta clase está encargada de preparar los Dialog Message, que serán las operaciones de pantalla que mostrarán el mensaje del usuario por pantalla cuando la alarma simple sea disparada.

Para un mayor conocimiento de las clases y sus métodos consultar el APÉNDICE F o el APÉNDICE D, para información sobre las bases de datos implantadas en esta plataforma. En cuanto al funcionamiento de la Alarma Simple, se basó en el concepto citado anteriormente del cual se obtuvo bastante documentación al respecto: las alertas de proximidad. Estas, son muy útiles para disparar una serie de instrucciones a seguir cuando el dispositivo móvil se encuentra cerca de algún punto en el mundo. Para un mayor conocimiento del funcionamiento de esta característica, se recomienda la consulta del APÉNDICE E. Se definieron los registros de las alertas de proximidad a lo largo de toda la aplicación, para que estuvieran preparadas para ser disparadas en todos los contextos de la misma.

Se definió finalmente la clase identificada anteriormente, "ProximityIntentReciver.java", en la cual se especificaron las acciones a realizar una vez disparada la alerta de proximidad. Esta clase representa un objeto del tipo BroadcastReceiver, el cual está asociado a una alerta de proximidad. Igualmente para más información, consultar el apéndice citado anteriormente.

Entre las instrucciones que se incluyen en la clase "ProximityIntentReciver.java", están la visualización del mensaje de alerta por pantalla y la configuración de notificaciones, que es otro medio para alertar a los usuarios del dispositivo móvil sin usar actividades o pantallas dentro de una aplicación. Son precisamente muy útiles estas notificaciones al momento de querer informar algo al usuario que requiere de su atención.

Usando la clase "Alerts.java", se definieron una serie de procedimientos para la configuración de los mensajes, para mostrar por pantalla el asunto definido en la alarma simple por el usuario. Aquí también es que se configuró la creación de las notificaciones, que son disparadas justo después de los mensajes de alerta. Esta clase es la encargada de ejecutar el tono que el usuario eligió para la alarma o activar el vibrador si así lo especificó. El mensaje de alerta se dispara de una manera simple, y se muestra por pantalla de una manera visible y de fácil entendimiento para el usuario. También se muestra la opción de "Desactivar" la alarma que fue disparada, para su eliminación de la base de datos, o simplemente "Cancelar" el mensaje una vez que ya fue visualizado pero que se quiere aún su disparo en próximas oportunidades.

Finalmente una vez ya codificado el Módulo de Gestión de Alarmas Simples, la codificación del Módulo Principal se vio afectada en la creación de nuevas opciones de menú. Se codificó un nuevo mensaje por pantalla, en el que el usuario pudiera elegir qué tipo de tarea quiere gestionar. A esta altura de la codificación se estableció las opciones, aún inactivas, para las futuras tareas que se implantarían en el sistema.

MÓDULO DE GESTIÓN DE RECORDATORIOS

Análisis de Requerimientos

Una vez definido el concepto de Alarma Simple, surgió la necesidad de Alarmas que no expiraran, y que tuvieran la capacidad de recordarle cierto asunto al usuario cada cierto tiempo. El usuario debiera ante esto asociar un mensaje a un recordatorio y especificar cada cuanto tiempo desea que éste sea mostrado en la aplicación. Esto sería muy útil si el usuario tiene alguna actividad que realizar cada cierto tiempo, tomarse la medicina, alimentar a las mascotas o simplemente recordar algo periódicamente hasta que éste decide que ya no vale la pena recordar ese asunto. Al igual que las alarmas simples, los Recordatorios deberían estar también asociados a ciertas coordenadas que el usuario especifique. Ante esto se propone la creación del Módulo de Gestión de Recordatorios, como segundo módulo asociado al Módulo de Gestión de Tareas por posicionamiento global de la aplicación a desarrollar.

Diseño global

El diseño contemplado para este módulo se asemeja en gran parte al ya implantado para el Módulo de Gestión de Alarmas Simples. El usuario debe ser capaz de poder registrar nuevos recordatorios cuando él lo requiera. Debe poder ingresar todos los datos asociados a ese recordatorio, y registrarlos relacionándolos a un par de coordenadas que ya existan en el Módulo de Registro de Coordenadas. Igualmente debe ser capaz de poder consultar los recordatorios cuando el requiere, y por consiguiente modificarlas o eliminarlos si así es requerido. El registro de recordatorios también debe estar completamente actualizado en todo momento en vista a las posibles modificaciones que pueda sufrir. Las salidas esperadas deben ser la visualización de mensajes de alerta en el dispositivo móvil perfectamente visibles para el usuario que lo hagan recordar el mensaje que éste especificó, cuando pase el radio de acción del lugar que definió en el registro. Asimismo, debe ser capaz el usuario de posponer el mensaje para que sea mostrado nuevamente después de un tiempo.

Diseño detallado

En el diseño detallado del Módulo de Gestión de Recordatorios se contempló la creación de una nueva pantalla para la aplicación. En esta pantalla el usuario tendrá los campos a rellenar de datos asociados al nuevo recordatorio que quiere almacenar en la base de datos de la aplicación.

Nombre del campo	Tipo de dato
Id	Integer
Asunto	String
Lugar Asociado	String
Radio de acción	Integer
Tipo de Alerta	String
Tiempo	String
Tiempo Exp	String
Posponer	String
Error	String

Tabla 5. Campos de la tabla "Recordatorio" de la Base de datos asociadas a la aplicación a desarrollar

Fuente: Elaboración propia

El asunto del recordatorio se planteó también como el mensaje que el usuario quiere recordar periódicamente con la alerta configurada. El lugar asociado, es el par de coordenadas en base al cual se disparará la alerta cuando se entre en el radio de acción, el cual también debe ser especificado por el usuario. Al igual que en las Alarmas Simples, se añadió un atributo para especificar el tipo de alerta que el usuario quisiera mostrar al disparar la alerta de proximidad. El usuario sería libre de elegir entre los tres tipos definidos: Vibrar, Tono o Vibrar + Tono, según sea su preferencia. El campo "Tiempo" del Recordatorio sería el tiempo que esperaría el sistema para volver a mostrar el Recordatorio, una vez que se haya pospuesto al mostrarse la primera vez. Después de analizar ciertas opciones, se decidió que el usuario decidiera entre las siguientes opciones predeterminadas en la aplicación: un (1) minuto, cinco (5) minutos, diez (10) minutos y treinta (30) minutos. El campo "Tiempoexp",

tendría más que todo una función a nivel de la aplicación: sería para saber, a nivel de sistema, en que tiempo se debería disparar el Recordatorio una vez que se haya pospuesto. Se tendría en cuenta el tiempo actual de sistema.

El campo "Posponer" se definió con el propósito de usarse como bandera que se active cuando un Recordatorio sea pospuesto la primera vez. Finalmente, también se estableció un campo de "Error" para el almacenamiento de algún tipo de error que se pueda presentar en el Recordatorio.

Para los recordatorios no se estableció un campo de expiración por tratarse de una alerta periódica. El usuario podrá posponerla hasta que desee, o hasta que la elimine de la base de datos al desactivarla. Al igual que las Alarmas Simples, el usuario tendrá la opción de registrar los recordatorios que desee, modificarlos cuando lo requiera, consultarlos cuando quiera y eliminarlos cuando deje de necesitarlos. En cuanto a las pantallas que definirían el diseño del Módulo de Gestión de Recordatorios, se contará con las siguientes:

- Pantalla de registro: el usuario debe ser capaz en esta pantalla de ingresar los datos asociados al recordatorio. En cuanto a las opciones de menú de la versión de ingresar de la pantalla se tiene: consultar o editar recordatorios, volver al mapa y salir de la aplicación. La versión de modificar permita adicionalmente eliminar el recordatorio que se está consultando.
- Pantalla de consulta: en esta pantalla el usuario consultaría los Recordatorios que hayan sido registrados en la aplicación con anterioridad. Esta pantalla ya fue diseñada e implementada para el Módulo de Gestión de Alarmas Simples, por ende sólo deberían hacerse las configuraciones necesarias para que sea adaptada para los Recordatorios. Al seleccionar algún Recordatorio dentro de la lista, la aplicación debería ir de inmediato a la pantalla de registro en su modo de modificar. Las opciones de menú de esta pantalla permitirán crear una nueva tarea, volver al mapa o salir de la aplicación.

El presente módulo hará interacción directamente con el Módulo Principal. Gracias al antiguo diseño del Módulo de Gestión de Alarmas Simples, ya el Módulo Principal cuenta con las opciones de menú implementadas las cuales deberían adaptarse también para el presente Módulo. Para consultas sobre el flujo de información, consultar el APÉNDICE C.

Codificación y depuración

Ya habiendo definido el diseño detallado, se comenzó a codificar el Módulo de Gestión de Recordatorios, estando finalmente representado, a nivel de desarrollo, por dos pantallas (una de ella ya desarrollada para el Módulo de Gestión de Alarmas Simples):

- PantallaRecord2.java: esta clase es la Pantalla de Registro del módulo. Presentará dos opciones de visualización. Si el usuario quiere registrar un nuevo recordatorio se podrá visualizar la pantalla en su versión de registro. Si en cambio, el usuario se encuentra en la Pantalla de Consulta, al seleccionar alguno de los recordatorios almacenados se podrá visualizar la versión de modificar. Esta clase estará relacionada con el archivo XML llamado "pantallarecord2.xml", que contendrá el código de la interfaz gráfica diseñada.
- PantallaConsultAlarm.java: esta clase fue implementada y codificada anteriormente para el Módulo de Gestión de Alarmas Simples. En esta clase, se visualiza una lista con todos los recordatorios que han sido registrados, mostrando el asunto del recordatorio y el lugar al que fue asociado.

Las capturas de las pantallas de la aplicación, así como sus casos de uso pueden ser consultados en el APÉNDICE H, adjunto al presente documento. Ya creadas las clases principales del módulo, el proceso de codificación continuó con el desarrollo de las clases que manipularían el objeto "Recordatorio", en cuanto a las operaciones de escritura y lectura. Para comenzar se creó la clase "Recordatorio.java", que representaría en la aplicación el objeto que se manipularía con todos los datos de un recordatorio. Esta clase contiene, después de haber evaluado el diseño implementado en la codificación, dos procedimientos: "RegistrarRecordatorios" y "RegistrarAlarmasRecordatorios". Los mismos son encargados de

hacer el registro de los recordatorios en toda pantalla de la aplicación, cuando no han sido disparados por primera vez y cuando son pospuestos, respectivamente. También una vez codificada la clase del objeto a manipular, se desarrollaron las clases que se encargarían de leer y escribir en la base de datos los Recordatorios así como los encargados de mostrarle al usuario el mismo cuando la alerta de proximidad del mismo haya sido disparada:

- RecordatorioDao.java: esta clase contiene todos los procedimientos y funciones que se encargarían de manipular la base de datos, en cuando al objeto Recordatorio se refiere.
- RecordatorioReceiver.java: es la clase que contiene la serie de pasos que se dispararán cuando el usuario haya entre en el radio de acción de un recordatorio la primera vez.
- RecordatorioRepetir.java: esta clase contiene la serie de pasos que deberán seguirse para mostrar un recordatorio una vez que éste fue pospuesto la primera vez.
- Alerts.java: esta clase ya fue definida para el Módulo de Gestión de Alarmas simples. Se agregaron nuevos procedimientos para preparar y mostrar los mensajes que contendrían el mensaje que el usuario quiere recordar.

Para consultar más detalladamente estas clases, se puede revisar adicionalmente el APÉNDICE F adjuntado o el APÉNDICE D, para información sobre las bases de datos implantadas en esta plataforma. Al igual que las Alarmas Simples, el concepto teórico básico en el que se basa el funcionamiento es el manejo de las alertas de proximidad. Para cada recordatorio se configuraría una alerta de proximidad. En cuanto a la expiración, sería infinita, es decir, jamás espiraría pues el usuario especificaría cuando debería desactivarse en el momento que él lo prefiera. Las instrucciones a ejecutarse cuando se dispare la alerta serán las contenidas en la clase "RecordatorioReceiver.java".

Los pasos que serían lanzados para la visualización, son los definidos en la clase "Alerts.java", fueron ya definidos desde la codificación del Módulo de Gestión de Alarmas Simples y algunos otros nuevos fueron añadidos. El procedimiento para la creación de notificaciones en el dispositivo móvil es usado también en el Módulo de Gestión de Recordatorios. Esta clase presenta dos nuevos procedimientos: uno para mostrar el

recordatorio la primera vez que es disparado y el segundo para mostrarlo una vez que es pospuesto la primera vez. La primera vez, el recordatorio es lanzado por pantalla, dando la opción al usuario de "Desactivar" el recordatorio, borrándolo de la base de datos, y también daría la opción de posponer el recordatorio por el tiempo que especificó en el registro. Una vez que es pospuesto el recordatorio, se eliminan las alertas de proximidad y se crean alarmas de tiempo. Estas alarmas de tiempo presenta un funcionamiento similar a las alertas de proximidad, y el mismo también puede ser consultado en el APÉNDICE E.

Los mensajes de alerta son mostrados de una manera estándar igualmente, independientemente si fueron pospuestos la primera vez. Se muestran por pantalla de una manera simple y de fácil visualización, siguiendo el formato utilizados para las alarmas simples.

Finalmente una vez ya codificado el Módulo de Gestión de Recordatorios, el Módulo Principal no fue afectado puesto que ya las opciones necesarias para interactuar con el nuevo.

MÓDULO DE GESTIÓN DE ALARMAS SMS

Análisis de Requerimientos

Al observar todos los recursos que provee Android, como plataforma, surgieron ideas sobre otras tareas que pudieran ayudar al usuario en su día a día. Uno de estos recursos es el envío de mensajes de texto de manera transparente para el usuario integrado con una alerta de proximidad de Android. De esta manera el usuario podrá asignarle a los lugares de su preferencia el envío de mensajes de texto de manera automática a la persona que él desee. Debido a esto se plantea la creación del Módulo de Gestión de Alarmas SMS como el tercer módulo asociado al Módulo de Gestión de Tareas por posicionamiento global de la aplicación a desarrollar.

Diseño global

El usuario debe ser capaz de acceder al menú y seleccionar una opción para crear la nueva alarma. Deberá introducir el destinatario al cual se enviará el mensaje, el texto del mismo, el lugar al cual estará asociado a la alarma, el radio de acción con el cual se disparará y finalmente un tiempo de expiración de la misma. Luego de introducir todos los datos, el usuario dejará registrada la alarma en el sistema. El mensaje de texto será enviado una vez que el usuario entre en el radio de acción alrededor del lugar seleccionado. Esta alarma será listada en la opción para consultar Alarmas SMS, donde estarán todas las alarmas creadas, y al seleccionarlas se podrá tener la opción de modificarlas o eliminarlas si así el usuario lo requiere. También el usuario deberá saber cuando sus mensajes no sean enviados, y cuáles son. Las salidas esperadas por el usuario al activarse la alarma deben ser la notificación de que el mensaje fue enviado al destinatario correctamente, y que el destinatario fue el que el usuario especificó previamente.

Diseño detallado

En el diseño detallado del Módulo de Gestión de Alarmas SMS se contempló la creación de nuevas pantallas para la aplicación. Al igual que en las alarmas anteriores, se trabajaría con alertas de proximidad de Android para la gestión del tipo de Alarmas SMS. Ante esto, se procedió a definir la estructura de datos que se manejaría en toda la aplicación y que estaría presente en la base de datos de la misma.

Nombre del campo	Tipo de dato
Id	Integer
Destinatario	String
Texto	String
Distancia	Integer
Lugar	Integer
Duración	Integer

Tabla 6. Campos de la tabla "Alarma Mensaje" de la base de datos asociada a la aplicación a desarrollar

Fuente: Elaboración propia

Apartando el identificador de la tabla, el primer campo de la tabla "Alarma Mensaje" es el "Destinatario", el cual almacena el número telefónico del destinatario al cual será enviado el mensaje de texto. El campo "Texto" seria el cuerpo del mensaje que el usuario podrá enviar en su mensaje. Luego está la "Distancia" que refiere al radio de acción que se le asignará a la alerta de proximidad de la alarma. El cuarto atributo es el "Lugar asociado", el cual poseerá el identificador del lugar en base al cual se configurará la alarma. Finalmente, se tiene la "Duración" que es el tiempo que la alarma estará vigente en el sistema, expresada en milisegundos.

En cuanto a las pantallas que definirían el diseño del Módulo de Gestión de Alarmas SMS, se crearía tres nuevas pantallas y se reutilizaría otras ya creadas:

- Pantalla de registro: el usuario debe ser capaz en esta pantalla de ingresar los datos asociados a la alarma. En su versión de ingresar, cuenta con opciones de menú que permiten consultar o editar alarmas SMS, volver al mapa o salir de la aplicación. En la versión de modificar, adicionalmente la aplicación permitirá eliminar la alarma consultada.
- Pantalla de consulta: en esta pantalla el usuario consultaría las Alarmas SMS que hayan sido registradas en la aplicación previamente. Esta pantalla ya fue diseñada e implementada para el Módulo de Gestión de Alarmas Simples. Al seleccionar alguna Alarma SMS dentro de la lista, la aplicación debería ir de inmediato a la pantalla de

registro en su modo de modificar. Las opciones de menú de esta pantalla permitirán crear una nueva tarea, volver a los mapas o salir de la aplicación.

- Pantalla de Mensajes no enviados: En esta pantalla se listarán todos los mensajes que no puedan ser enviados al momento de ser disparada la alarma.
- Pantalla para Reenviar mensajes: en esta pantalla estarán todos los datos un mensaje no enviado cargados, donde el usuario podrá editarlos y reenviar el mensaje.

El presente módulo hará interacción directa con el Módulo Principal de la aplicación. Gracias al antiguo diseño del Módulo de Gestión de Alarmas Simples, ya el Módulo Principal cuenta con las opciones de menú implementadas, sólo se añadiría una nueva perteneciente a la consulta de mensajes no enviados. En el APÉNDICE C, se puede consultar a cabalidad el flujo de información de toda la aplicación.

Codificación y depuración

Teniendo definido el diseño detallado, se comenzó con la codificación del Módulo de Gestión de Alarmas SMS, el cual es finalmente representando por cuatro pantallas, codificadas en cuatro clases:

- PantallaM.java: esta clase es la Pantalla de Registro del módulo. Presentará dos opciones de visualización: si el usuario quiere registrar una nueva Alarma SMS se podrá visualizar la pantalla en su versión de registro. Si en cambio, el usuario se encuentra en la Pantalla de Consulta, al seleccionar alguno de las Alarmas SMS almacenadas se podrá visualizar la versión de modificar de esta clase. Igualmente, esta clase estará relacionada con el archivo XML llamado "pantallam.xml", que representará el interfaz de la pantalla.
- PantallaConsultAlarm.java: esta clase fue implementada y codificada anteriormente para el Módulo de Gestión de Alarmas Simples. En esta clase, se visualiza una lista con todas las Alarmas SMS que han sido registradas, mostrando la alarma mensaje y el lugar al que fue asociado.

- PantallaMnE.java: esta clase es la pantalla que contiene la lista de mensajes no enviados. Al seleccionar un mensaje de la lista, el mensaje se podrá editar o reenviar.
- PantallaMR.java: esta clase es la pantalla que contiene los datos de un mensaje no enviado. Presenta los datos del mensaje y el usuario los podrá editar y reenviar el mensaje pulsando un botón.

Las capturas de las pantallas de la aplicación, así como sus casos de uso pueden ser consultados en el APÉNDICE H, adjunto al presente documento. El proceso de codificación continuó con el desarrollo de las clases que manipularían el objeto "Alarma Mensaje". Para comenzar se creó la clase "AlarmaMensaje.java", que representaría en la aplicación el objeto que se manipularía con todos los datos de una Alarma SMS. También se desarrollaron todas las clases que se encargarían de leer y escribir en la base de datos las Alarmas SMS así como los encargados de mostrarle al usuario el recordatorio cuando la alerta de proximidad del mismo haya sido disparada. Finalmente las clases resultantes fueron las siguientes:

- AlarmaMensajesDao.java: esta clase contiene todos los procedimientos y funciones que se encargarían de manipular las alarmas registradas en la base de datos.
- AlarmaMensajesReceiver.java: esta clase contiene la serie de pasos que se dispararán una vez que el usuario haya entrado en el radio de acción de una Alarma SMS.
- EstadoMensajeReceiver.java: esta clase contiene la serie de pasos que se dispararán una vez que sea enviado el mensaje de la alarma de proximidad inicial de la Alarma SMS. Se encargará de avisar si el mensaje fue entregado o no y porque razón.

Para consultar más detalladamente estas clases, revisar el APÉNDICE F o el APÉNDICE D, para información sobre las bases de datos implantadas en esta plataforma. Al igual que las Alarmas Simples y los Recordatorios, el concepto teórico básico en el que se basa el funcionamiento es el manejo de las alertas de proximidad de Android, las cuales constan de disparadores que son activados una vez que el dispositivo móvil entra en un radio de acción especificado alrededor de un lugar determinado. Para una consulta detallada del

funcionamiento de las alertas de proximidad, se recomienda hacer referencia al APÉNDICE E, adjuntado al presente documento.

Como estas alarmas en específico envían mensajes de texto, se debe listar una serie de instrucciones adicionales que estén pendientes del estado del mensaje que ha intentado enviar. En este caso, estamos hablando de la clase "EstadoMensajeReceiver.java", el cual estaría emparentado con mediantes claves iguales con el objeto "AlarmaMensajesReceiver". Todos los mensajes que no puedan ser enviados por el teléfono, podrá ser accedidos a través de la opción del menú principal "Mensajes no enviados (AM)", donde el usuario podrá seleccionar sus mensajes y enviarlos de nuevo. También una explicación bastante amplia de este funcionamiento, se encuentra documentada en el APÉNDICE E. Al dispararse las alarmas, el usuario será notificado a través de un mensaje instantáneo por pantalla, y una notificación que el mensaje fue enviado o no y por cual razón.

MÓDULO DE GESTIÓN DE ALARMAS EMAIL

Análisis de Requerimientos

Analizando todas las posibilidades que provee Android, como plataforma, surgieron ideas sobre una alarma más que pudiera ayudar al usuario en vida cotidiana. Esto hace referencia al envío de correos electrónicos de manera transparente. De esta manera el usuario podrá asignarle a los lugares de su preferencia el envío de correos de manera automática a la persona que él desee. Así se podrá configurar previamente esta tarea y el usuario no deberá preocuparse por el envío del correo. Debido a esto, se plantea la creación del Modulo de Gestión de Alarmas Email como el cuarto módulo asociado al Módulo de Gestión de Tareas por posicionamiento global de la aplicación a desarrollar.

Diseño global

El usuario debe ser capaz de acceder al menú y seleccionar la opción para crear una Alarma Email, la cual le permitirá ir a la pantalla de configuración del correo electrónico asociado. Aquí el usuario deberá introducir los datos asociados a la alarma. Luego de introducir todos los datos el usuario deberá registrar finalmente la alarma en la base de datos para que el correo sea enviado, una vez que el usuario se acerque la distancia seleccionada al lugar determinado. El usuario deberá poder consultar las alarmas creadas y al seleccionarlas se tendrá la opción de modificarlas o eliminarlas. También el usuario debería saber qué correos no fueron enviados por cualquier razón. Las salidas esperadas deben ser la notificación de que el correo fue enviado correctamente, justo en el momento en el que el dispositivo entre en el radio de acción de la alarma configurada en el sistema.

Diseño detallado

En el diseño detallado del Módulo de Gestión de Alarmas Email se contempló la creación de nuevas pantallas para la aplicación. Al igual que en las alarmas anteriores, se trabajaría con alertas de proximidad de Android para la gestión del tipo de Alarmas Email. Ante esto, se procedió a definir la estructura de datos que se manejaría en toda la aplicación y que estaría presente en la base de datos de la misma:

Nombre del campo	Tipo de dato
Id	Integer
Destinatario	String
Asunto	String
Texto	String
Distancia	Integer
Lugar	Integer
Duración	Integer

Tabla 7. Campos de la tabla "Alarma Correo" de la Base de datos asociadas a la aplicación a desarrollar

Fuente: Elaboración propia

Como todas las alarmas de la aplicación son basadas en alertas de proximidad, poseen la misma estructura. Apartando el identificador de la tabla, el primer campo de la tabla "Alarma Correo" es el "Destinatario", el cual representa la dirección de correo electrónico del destinatario al cual será enviado el email configurado. El siguiente campo es el "Asunto" del correo, que se refiere al título del correo en cuestión. También está el cuerpo o "Texto" del mensaje en el cual el usuario podrá escribir el texto que desee enviar adjuntado al correo. Luego está la "Distancia" que es el radio de acción que se le asignará a la alerta de proximidad para que envíe el correo electrónico especificado. El quinto atributo es el "Lugar" asociado, el cual poseerá el identificador del lugar con el que está asociado la alerta de proximidad. Finalmente, se tiene la "Duración" que es el tiempo que la alarma estará vigente en la aplicación, expresada en milisegundos.

En cuanto a las pantallas que definirían el diseño del Módulo de Gestión de Alarmas Email, se contempla la creación de tres nuevas pantallas y se reutilizaría otras ya creadas:

- Pantalla de registro: el usuario debe ser capaz en esta pantalla de ingresar los datos asociados a la alarma. En su versión de ingresar cuenta con opciones de menú que permiten consultar o editar alarmas Email, volver al mapa o salir de la aplicación. En la versión de modificar, adicionalmente la aplicación permitirá eliminar la alarma consultada.
- Pantalla de consulta: en esta pantalla el usuario consultaría las Alarmas Email que hayan sido registradas en la aplicación. Esta pantalla ya fue diseñada e implementada para el Módulo de Gestión de Alarmas Simples. Al seleccionar alguna Alarma Email dentro de la lista, la aplicación debería ir de inmediato a la Pantalla de Registro en su modo de modificar, dando la opción al usuario de actualizar los datos de una Alarma Email registrada. Las opciones de menú de esta pantalla permiten crear una nueva tarea, volver a los mapas y salir de la aplicación.
- Pantalla de Correos no enviados: en esta pantalla se listarán todos los correos no enviados para que el usuario pueda reenviarlos cuando desee. Al seleccionar algún correo, el usuario lo podrá editar y posteriormente enviar.

- Pantalla reenviar correos: en esta pantalla estarán cargados todos los datos del correo no enviado donde el usuario podrá modificarlos y posteriormente enviarlo.

El presente módulo hará interacción directamente con el Módulo Principal. Gracias al antiguo diseño del Módulo de Gestión de Alarmas Simples, ya el Módulo Principal cuenta con las opciones de menú implementadas. Sólo se agregaría una nueva opción asociada a los correos no enviados. En el APÉNDICE C, se contempla documentado el flujo de información de toda la aplicación y de sus módulos de gestión.

Codificación y depuración

Contando con el diseño detallado del Módulo De Gestión de Alarmas Email, se comenzó con el proceso de codificación. El mismo, está representado por cuatro pantallas (una de ella ya desarrollada para el Módulo de Gestión de Alarmas Simples):

- PantallaC.java: esta clase es la Pantalla de Registro del módulo. Presentará dos opciones de visualización: si el usuario quiere registrar una nueva Alarma Email, se podrá visualizar la pantalla en su versión de registro. Si en cambio, el usuario se encuentra en la Pantalla de Consulta, al seleccionar alguno de las Alarmas Email almacenadas se podrá visualizar la versión de modificar de esta clase. Igualmente, esta clase estará relacionada con el archivo XML llamado "pantallac.xml", que contiene la el código de interfaz gráfica.
- PantallaConsultAlarm.java: esta clase fue implementada y codificada anteriormente para el Módulo de Gestión de Alarmas Simples. Finalmente, aquí se visualiza una lista con todas las Alarmas Email que han sido registradas, mostrando la alarma correo y el lugar al que fue asociado.
- PantallaCnE.java: esta clase representa la pantalla de correos no enviados. En ella estará una lista donde el usuario podrá seleccionar el correo que desea reenviar. Al seleccionar un correo este será reenviado a otra actividad con los datos del correo.

- PantallaCR.java: esta clase representa la pantalla de reenvío de correos. Presentará los campos del correo a reenviar, los cuales serán el destinatario el asunto y el cuerpo del correo. Una vez editados y confirmados, se ejecutara el envió del correo nuevamente.

Las capturas de las pantallas de la aplicación, así como sus casos de uso pueden ser consultados en el APÉNDICE H, adjunto al presente documento. El proceso de codificación continuó con el desarrollo de las clases que manipularían el objeto "Alarma Correo", en cuanto a las operaciones de escritura y lectura. Para comenzar se creó la clase "AlarmaCorreo.java", que representaría en la aplicación el objeto que se manipularía con todos los datos de una Alarma Email, dando la opción de poder manipularse dinámicamente. También una vez codificada la clase del objeto a manipular, se desarrollaron las clases que se encargarían de leer y escribir en la base de datos las Alarmas Email así como los encargados de mostrarle al usuario los mensajes respectivos cuando la alerta de proximidad del mismo haya sido disparada. Finalmente las clases resultantes fueron las siguientes:

- AlarmaCorreoDao.java: esta clase contiene todos los procedimientos y funciones que se encargarían de manipular la base de datos, en cuando al objeto "AlarmaCorreo" se refiere.
- AlarmaCorreoReceiver.java: esta clase contiene la serie de pasos que se dispararán una vez que el usuario haya entrado en el radio de acción de una Alarma Email.

Una descripción de todas estas clases puede ser consultada con más detalle en el APÉNDICE F o el APÉNDICE D, para información sobre las bases de datos implantadas en esta plataforma. Todas las clases especificadas anteriormente definen lo que sería el funcionamiento de las Alarmas Email para el módulo en cuestión. Al igual que las Alarmas Simples, el concepto teórico básico en el que se basa el funcionamiento es el manejo de las alertas de proximidad. Para cada Alarma Email que cree el usuario se creará una alerta de proximidad. El funcionamiento de éstas, se describe en el APÉNDICE E, disponible para su consulta. Al dispararse las alarmas el usuario será notificado a través de un mensaje

instantáneo y una notificación si el correo fue enviado con éxito o no. De no ser enviados serán listados en la opción del menú principal "Correos no enviados (AC)".

MODULO DE CONSULTA DE TRÁFICO

Análisis de Requerimientos

Luego de realizar en el Módulo de Gestión de Tareas una alarma que es capaz de enviar un mensaje de texto al contacto que se desee, surgió la idea de poder enviar información por medio de un mensaje de texto, cuando el usuario entre en una vía muy transitada de Caracas, a un servidor y de esta manera poder realizar operaciones sobre estos datos para saber cuánto tiempo tardó un usuario en específico en llegar de un lugar a otro. El servidor podría recibir el mensaje almacenarlo para su posterior consulta por parte de otros usuarios que deseen saber el tiempo que tiene registrado el servidor entre dos lugares en específico. De esta manera un usuario contribuye a crear la base de datos en tiempo real y también podrá consultarla posteriormente.

Diseño global

El usuario debe ser capaz de acceder a una ventana de consulta de tráfico y allí deberá registrarse en el Módulo de Consulta de Tráfico. Deberá proporcionar su tipo de vehículo y su número de teléfono. Posteriormente debe seleccionar el servidor contra el cual desea se hará la consulta. Luego de seleccionar el servidor aparecerá una pantalla donde deberá seleccionar el nodo de origen de la ruta que desea consultar. Una vez seleccionado deberá ser capaz de tener dos opciones:

1. Consultar rutas: esta opción llevará al usuario a una pantalla con todas las rutas registradas en el sistema que partan de ese nodo. Aquí el usuario seleccionará una de las rutas registradas en el sistema para consultar tráfico. De esta manera se podrá enviar un

mensaje de consulta al servidor. En los próximos segundos el usuario recibirá por parte del servidor un mensaje de texto con el tiempo mínimo registrado para esta ruta.

2. Consultar rutas: esta opción llevará a una pantalla donde aparecerán los nodos destino a los cuales se puedan llegar por varias rutas partiendo de un mismo nodo origen. De esta manera el usuario seleccionará un nodo origen y un nodo destino y recibirá por parte del servidor un mensaje ordenado de menor a mayor por tiempo, de todas las rutas existentes entre esos dos nodos.

Diseño detallado

En el diseño detallado del Módulo de Gestión de Trafico se contempló la creación de cinco pantallas para la aplicación. En la primera pantalla el usuario tendrá que registrarse colocando su número telefónico y su tipo de vehículo. En la segunda pantalla deberá seleccionar un servidor contra el cual hará la consulta y en la tercera pantalla tendrá que seleccionar el nodo de origen desde el cual desea realizar la consulta. Este nodo filtrará la información para las otras dos pantallas, ya que en una mostrará todas las rutas posibles que parten de este nodo y en la otra todos los nodos posibles al que puede llegar ese nodo. Ante esto, se crearon nuevas estructuras de datos necesarias para el manejo de datos.

Objeto	Campos	Tipos de dato
Ruta	o Id o Referencia	o Integer o String
Nodo	o Id o Nombre o Latitud o Longitud o Clase o Rango	o Integer o String o String o String o String o Integer
Ruta-Nodo	o IdNodo o IdRuta o Sentido Ruta	o Integer o Integer o Integer
Registro_Usuarios	o Id o NumeroCelular o TipoVehiculo	o Integer o String o String
Servidores	o Id o NumeroCelular	o Integer o String

Tabla 8. Tablas de la base de datos asociada al Módulo de Consulta de Tráfico

Fuente: Elaboración propia

- "Ruta": representa las rutas registradas para el Módulo de Gestión del Tráfico, aquí se guarda el identificador de la ruta y la referencia, la cual es una explicación a base de puntos de referencia acerca de la misma.
- "Nodo": representa los nodos que estén registrados. Los atributos son: el identificador único, el nombre del nodo, la latitud y la longitud para ubicar al nodo, el tipo de nodo (entrada o salida) y el rango o radio de acción único para cada nodo dependiendo de su tamaño en la vía.
- "Ruta-Nodo": representa una relación doble entre ruta y nodo, contiene el identificador de la ruta, el del nodo y un sentido de ruta. El sentido de ruta indica que parte de la ruta es cada nodo, es decir, a cada nodo se le asigna un número formando una secuencia que muestra la posición de cada nodo en la ruta.

- "Registro_usuarios": representa los usuarios registrados en el Módulo.
- "Servidores": representa los servidores disponibles para realizar las consultas de tráfico.

En cuanto a las pantallas del Módulo de Gestión de Trafico se definieron cinco:

- Pantalla de Registro de usuario: en esta pantalla el usuario debe ser capaz seleccionar un tipo de vehículo e introducir su número de teléfono. Una vez introducidos todos los datos, el usuario deberá ser capaz de pulsar un botón para ir a la pantalla de selección de servidor. Esta operación se hará al iniciar la aplicación por primera vez. Las opciones de menú de esta pantalla contarán con un regreso a los mapas y la opción de salir de la aplicación.
- Pantalla de Consulta de servidor: en esta pantalla el usuario debe ser capaz de seleccionar un servidor de mensajes de texto contra el cual hará la consulta de tráfico. Una vez seleccionado deberá ser capaz de pulsar un botón para ir a la pantalla de consulta de nodo. Las opciones de menú de esta pantalla contarán con un regreso a los mapas y la opción de salir de la aplicación.
- Pantalla de Consulta de nodo: en esta pantalla el usuario debe ser capaz de seleccionar un nodo de origen desde el cual desea hacer la consulta para luego ser capaz de pulsar dos botones: uno para consultar todas las rutas registradas que parten de ese nodo, y otro para consultar todas las rutas existentes entre dos nodos. Las opciones de menú de esta pantalla contarán con un regreso a los mapas y la opción de salir de la aplicación.
- Pantalla de Consulta de rutas: en esta pantalla el usuario debe ser capaz de seleccionar una de las rutas que parten del nodo origen seleccionado. Una vez seleccionado, el usuario debe pulsar un botón para realizar la consulta de tráfico. La aplicación enviará un mensaje al servidor y éste le responderá al usuario con un mensaje de texto con la consulta. Las opciones de menú de esta pantalla contarán con un regreso a los mapas y la opción de salir de la aplicación.
- Pantalla de consulta de nodos salida: en esta pantalla el usuario debe ser capaz de seleccionar un nodo de salida que este asociado con el nodo de entrada seleccionado.

Una vez seleccionado, el usuario deberá ser capaz de pulsar un botón para realizar la consulta de tráfico el cual enviará un mensaje con la consulta al servidor y este responderá con un mensaje con todas las rutas posibles entre esos dos nodos. Las opciones de menú de esta pantalla contarán con un regreso a los mapas y la opción de salir de la aplicación.

Codificación y depuración

Luego de definir el diseño detallado, se comenzó a implantar el Módulo de Gestión de Tráfico, el cual cuenta con cinco (5) pantallas:

- RegistroUsuario.java: en esta pantalla el usuario deberá introducir su número telefónico y su tipo de vehículo, luego deberá ser capaz de pulsar el botón "Continuar consulta" para poder seleccionar el servidor.
- ConsultaServidor.java: en esta pantalla el usuario tendrá todos los servidores disponibles, así podrá seleccionar el servidor que desee. Luego de seleccionarlo deberá ser capaz de pulsar el botón "Continuar consulta", el cual llevará al usuario a la siguiente pantalla.
- ConsultaTrafico.java: esta es la pantalla de consulta inicial. En ésta el usuario tendrá todos los nodos origen registrados en la aplicación. De esta manera podrá seleccionar el nodo desde el cual desea hacer la consulta y posteriormente pulsar uno de los dos botones que continuarán con la consulta. El primer botón sería "Consultar rutas", el cual llevaría al usuario a otra pantalla con todas las rutas que salen de ese nodo, y el segundo botón sería "Consultar nodos destino", el cual llevaría al usuario a otra pantalla con todos los nodos destino a donde puede llegar el nodo de entrada.
- ConsultaTraficoRutas.java: esta pantalla muestra al usuario todas las rutas existentes que parten del nodo origen seleccionado por el usuario en la pantalla anterior. Una vez seleccionada la escogida, se podrá pulsar el botón "Realizar Consulta de Tráfico", el cual enviará el mensaje al servidor de tráfico con la consulta del usuario.

Las capturas de las pantallas de la aplicación, así como sus casos de uso pueden ser consultados en el APÉNDICE H, adjunto al presente documento. Una vez creada las clases

principales, se procedió a crear las clases que manipularán a los objetos Nodo, Ruta y Servidor en cuanto a operaciones de lectura y escritura de la base de datos se refiere. Las clases desarrolladas fueron las siguientes:

- NodoDao.java: esta clase contiene todos los métodos encargados de manipular los datos del objeto Nodo.
- RutaDao.java: esta clase contiene todos los métodos encargados de manipular los datos del objeto Ruta.
- ServidorDao.java: esta clase contiene todos los métodos encargados de manipular los datos del objeto Servidor.

Todas las clases definidas anteriormente completan las clases necesarias para el funcionamiento del Módulo de Tráfico de la aplicación, y pueden ser consultadas con mayor detalle en el APÉNDICE F o el APÉNDICE D, para información sobre las bases de datos implantadas en esta plataforma. Este módulo ejecuta una consulta contra un servidor de *mensajes de texto el cual responderá con otro mensaje con la respuesta de la consulta* realizada. Al igual que el usuario puede realizar consultas para saber el estado del tráfico sobre el servidor, él también colabora con el mismo de manera automática. Cuando el usuario circule con su vehículo por las vías mapeadas, la aplicación enviará un mensaje de texto al servidor con la posición del usuario, fecha y hora, para que de esta manera se guarde un registro sobre el que luego se puedan enviar consultas.

Este envío de mensajes se realiza a través de una alerta de proximidad que es creada sobre las entradas y salidas de las vías de tráfico. Como se ha recomendado antes, se puede consultar el APÉNDICE E para un mayor conocimiento de las alertas de proximidad de Android. Como las alertas de proximidad de tráfico no serán creadas en tiempo de ejecución por el usuario, están serán registradas en el AndroidManifest.xml, de esta manera al ser creadas al iniciar la aplicación estas quedan registradas automáticamente. Una vez registradas el usuario colaborara con el servidor cada vez que pase por alguna entrada o salida de las rutas registradas en el teléfono. Esto será transparente para el usuario. Él

colaborará de forma automática y podrá consultar esa base de datos las veces que desee a través del Módulo de Gestión de Tráfico. Este módulo construye un mensaje de texto con las opciones que va seleccionando el usuario: primero coloca el encabezado y luego la ruta o los nodos que desea consultar el usuario. Una vez que el usuario selecciona su consulta y construido una cadena de caracteres para el mensaje, se envía un mensaje de texto al servidor seleccionado.

MÓDULO DE SERVIDOR DE MENSAJES DE TEXTO

Análisis de Requerimientos

Una vez creado el modulo de gestión de tráfico en la aplicación de Android, se procedió con la implementación del Servidor de mensajes de texto, el cual estará encargado de recibir y enviar mensajes de registro y de consulta de tráfico.

Diseño global

El servidor de texto está conformado por un servidor GSM y un entorno con una librería de java. El modem GSM utilizado fue un Motorola C650 con formato texto, y en el entorno corre un programa en Java que utiliza una librería para la gestión de mensajes de texto a través del lenguaje. De esta manera se utilizará el modem GSM para la recepción y envío de los mensajes de texto y a través de la programación Java se podrá acceder al texto que contienen y construir y enviar mensajes de texto al número que se desee. Inicialmente el modem GSM debe estar conectado al computador. Ésta conexión será a través de un cable USB. Una vez conectado, se correrá la aplicación escrita en Java. El servidor puede recibir mensajes de tipo consulta o de tipo registro, se sabrá el tipo de mensaje. Si el mensaje es del tipo consulta se debe enviar un mensaje de texto, si es de registro el servidor debe almacenar la información del mismo en la base de datos.

Diseño detallado

El servidor de mensajes de texto siempre estará activo para la recepción y envió de mensajes de texto. Para el correcto funcionamiento del servidor se crearon las siguientes estructuras de datos, además de rehusar los conceptos de Ruta, Nodo, Ruta-Nodo y Registro_Usuarios, desarrollados anteriormente:

Tabla	Campos	Tipos de dato
Historial_entradas	o Id o Fecha o Nodo o NumeroCelular	o Integer o TimeStamp o Integer o String
Historial_salidas	o Id o Fecha o Nodo o NumeroCelular	o Integer o TimeStamp o Integer o String
Entradas_salidas	o Id_historial_entradas_f o Id_historial_salidas_f	o Integer o Integer
Tiempos_estimados	o Id o Id_nodo_origen_f o Id_nodo_destino_f o Tiempo_estimado_no_pico o Tiempo_estimado_pico	o Integer o Integer o Integer o Integer o Integer

Tabla 9. Tablas de la base de datos relacionadas con el Módulo del Servidor de mensajes de texto

Fuente: Elaboración propia

En la tabla Historial_entradas e Historial_salidas se guardarán todos los datos de los mensajes del tipo Entrada y salida respectivamente. Cuando el servidor reciba un mensaje de tipo entrada o salida, este será desfragmentado para obtener cada registro y guardarlo en la tabla. También se tiene el identificador único del mensaje, la fecha en la cual se envió el

mensaje, el identificador del nodo de entrada desde el cual se realizó el envío y el número del celular del usuario.

La tabla Tiempos_estimados contiene tiempos promedios de cada ruta que el servidor utilizará en caso de no exista ningún registro de las rutas o para comparar y evitar registros atípicos. Contiene el identificador único de la tabla, el identificador del nodo origen de la ruta, el identificador del nodo destino de la ruta, el tiempo que toma la ruta en horas de mucho tráfico y el tiempo que toma la ruta en tiempos de poco tráfico.

Codificación y depuración

El servidor de mensajes de texto cuenta con tres paquetes principales:

1. Lógica: en este paquete se encuentran las clases que poseerán toda la lógica de negocio del servidor. Las clases que posee este paquete son las siguientes:
    a. "Servidor.java": es la clase principal del servidor. Está encargada de recibir y enviar mensajes, además de su desfragmentación. Implementa una interfaz de la librería de envío de mensajes de Java, la cual es llamada "NotificacionNuevosSms", que permite sobrescribir los métodos de la librería para realizar acciones al momento de recibir mensajes.
2. Clases: en este paquete se encuentran todas las clases que manipularán los datos del servidor de mensajes de texto. Las clases son las siguientes:
    a. "MensajesConsulta.java": está encargara de manipular todo los datos de los mensajes del tipo consulta.
    b. "MensajesRegistroEntrada": esta clase se encarga de manipular todos los datos relacionados con los mensajes del tipo entrada del servidor.
    c. "MensajesRegistroSalida": está encargada de gestionar todo lo relacionado con los mensajes del tipo salida que recibe el servidor.
    d. "MensajesRegistroUsuario.java": está encargada del registro de los usuarios que utilicen el servidor.

Una vez definidas todas las clases se definieron todas las clases que gestionaran la lectura y escritura de los datos en las clases:

- MensajesConsultaDao.java: esta es la clase encargada de gestionar todos los datos relacionados con los objetos del tipo "MensajesConsulta", en cuanto a sus operaciones de lectura y escritura.
- *MensajesRegistroEntradaDao.java: esta clase es la encargada de gestionar todos los* datos relacionados con los objetos del tipo MensajesRegistroEntrada, en cuanto a sus operaciones de lectura y escritura.
- MensajesRegistroSalidaDao.java: esta clase es la encargada de gestionar todos los datos relacionados con los objetos del tipo MensajesRegistroSalida, en cuanto a sus operaciones de lectura y escritura.
- MensajesRegistroUsuarioDao.java: esta clase está encargada de gestionar todos los datos relacionados con los objetos del tipo "MensajesRegistroUsuario", en cuanto a sus operaciones de lectura y escritura.

Además de las clases antes explicadas, se realizo un procedimiento almacenado en la base de datos implementada llamado "TIEMPO_NODO_E_S", el cual se encarga de calcular el tiempo entre dos nodos. Es utilizado por los métodos de las clases que manipulan las operaciones de lectura y escritura. De esta manera se puede calcular el tiempo entre dos nodos llamando a este procedimiento y pasando por parámetro los dos nodos en cuestión.

Todas las clases definidas anteriormente son las necesarias para el correcto funcionamiento del servidor de mensajes de texto. El servidor como se explico anteriormente tiene tres (4) funciones básicas: registrar mensajes de nodos entrada, registrar mensajes de nodo salida, registrar usuarios y responder consultas que realicen los usuarios al servidor.

Al arrancar el servidor este inicializa un hilo, el cual captura todos los mensajes que lleguen al servidor, y ejecuta un método llamado NuevoSMS(). Una vez inicializado el hilo, el servidor queda activo a la espera de mensajes de texto. Al recibir un mensaje de texto el servidor lo desfragmenta pues llegan con un formato especifico para la aplicación. Este

mensaje está formado por varios registros. El primer registro es el que indica que tipo de mensaje es: uno (1) para los mensajes de tipo consulta, dos (2) para los mensajes de tipo salida, tres (3) para los mensajes de tipo entrada y cuatro (4) para los mensajes de tipo registro usuario. Dependiendo de este valor el servidor realizará acciones diferentes.

Para los mensajes del tipo registro Entrada se crea un objeto "MensajesRegistroEntrada". Al llegar el mensaje se chequera si el usuario está registrado y si su tipo de vehiculo es "Automóvil". De ser asi se llenan todos los atributos del objeto con los datos del mismo, y se llama al método del objeto "insertarMensajeEntrada", al cual se le pasa por parámetro un objeto de tipo "MensajeRegistroEntrada" con todos los datos. Sólo se toman los registros de Automóviles ya que las motos no serán contempladas para esta entrega. El método se conecta con la base de datos e inserta los datos en la tabla Historial_entradas, donde se guarda el identificador del nodo, la fecha y el número de celular del cual fue enviado.

Para los mensajes del tipo registro Salida se crea un objeto "MensajesRegistroSalida". Se chequea que el usuario tenga automóvil y todos los atributos del objeto son llenados con los datos del mensaje para posteriormente llamar al método "insertarMensajeSalida". Este método inserta en la tabla Historial_entradas el identificador del nodo, la fecha de registro y el número de celular del usuario. En este caso no sólo se debe insertar en esta tabla, también en la relación entre la relación doble entre las entradas y salidas. Para esto se busca en la tabla "Historial_entradas" el registro más actual que posea el número de celular con el cual llegó el mensaje de salida. Al conseguir este registro se consigue la entrada de donde proviene el mensaje de salida y se inserta en la base de datos. Con esto ya se tiene asignada una entrada a una salida, necesario para consultas posteriores.

Para los mensajes del tipo Registro de usuario, el servidor crea un objeto del tipo "MensajesRegistroUsuario", y llama al método "insertarUsuario". Se llenan los atributos del objeto con todos los datos del mensaje de texto y es pasado por parámetro al método para que inserte el usuario en la base de datos.

Para los mensajes del tipo Consulta, el servidor crea un objeto de tipo "MensajesConsulta", y llama al método "consultaTrafico". Al crear el objeto se llenan los nodos involucrados, los cuales representan el origen y el fin de la ruta a consultar. El método "consultaTrafico" llama a "MensajesConsultaDao" y se conecta con la base de datos. Lo primero que hace el método es buscar en la tabla "Ruta" todas las rutas que empiecen por el nodo origen. Una vez obtenido esto, se procede a buscar todos los nodos de cada ruta y se chequea el nodo final de cada una. El mensaje de consulta puede ser de dos tipos, uno es consultar una ruta en específico seleccionada por el usuario y otra es consultar todas las rutas disponibles entre dos nodos.

Una vez que se tienen todas las rutas se toman los nodos que pertenecen a ella y se empieza a calcular el tiempo que hay entre cada nodo. Este tiempo se va sumando en una variable hasta que se llegue al nodo final de la ruta. Para el cálculo del tiempo se utiliza un procedimiento almacenado debido a que al incrementar el número de usuarios, en un futuro, la cantidad de registros en la base de datos será muy grande, por lo cual es más rápido tener un procedimiento que calcule el tiempo y lo devuelva al servidor. El procedimiento almacenado tiene dos tipos de búsqueda, uno entre nodos entrada-salida, y otro entre nodos salida-entrada. Esto se debe a que una ruta tiene la secuencia entrada-salida-entrada-salida y así sucesivamente.

Si el tipo de consulta es de una ruta, se calcula todo el tiempo que hay entre cada nodo y se guarda en una variable "tiempo". El servidor en envía un mensaje con el tiempo calculado y la fecha al usuario. En caso de ser el otro tipo de búsqueda, el servidor calcula el tiempo de todas las rutas y va concatenando cada una a una cadena de caracteres que luego será enviando en un mensaje al usuario.

Cada nodo perteneciente a una ruta posee un sentido, es decir, su posición en la ruta, de manera que se va calculando el tiempo entre cada nodo guiándose por su sentido. De esta manera se va calculando a través del procedimiento almacenado el tiempo que hay entre cada nodo de una ruta en específico hasta llegar a un tiempo final que es enviado al usuario.

Puede darse el caso de que el servidor no posea registros sobre una consulta que realice el usuario, ya sea porque ningún usuario ha pasado por la misma o porque el registro sea muy viejo, para ello se tiene una tabla de tiempos estimados. Esta tabla posee tiempos promedios de todas las rutas registradas en horas de mucho y poco tráfico, por lo cual en caso de que el usuario haga una consulta y el servidor no posea datos actuales, este consultará la tabla de tiempos estimados y se lo hará saber al usuario. También es importante resaltar que puede darse el caso de que un usuario se salga en un nodo en una ruta registrada, y se detenga en algún punto entre el nodo salida y entrada a realizar alguna diligencia. Supóngase también que este es el único registro que posee el servidor, por lo cual si un usuario realiza una consulta el servidor devolverá un tiempo que no es real ya que no estará tomando en cuenta el tiempo que el usuario perdió detenido en su diligencia. Para mitigar estos riesgos al máximo, cada vez que se calcula un tiempo, este es comparado con el tiempo estimado, si este tiempo es mucho mayor al estimado se toma como valor atípico y no es tomado en cuenta, y se utiliza el tiempo estimado para la consulta del usuario.

ANDROID FRENTE A OTRAS PLATAFORMAS
ESTUDIO COMPARATIVO

Android llegó a un mercado en el que actualmente se debe enfrentar con muchas tecnologías y plataformas que se han encargado de presentarse como las mejores en el campo de la telefonía móvil. LiMo, Symbian y OS X Mobile se han sabido colar en el mercado, y hoy en día las tres prometedoras tecnologías son buscadas por grandes cantidades de usuarios alrededor de todo el mundo. Tomando en cuenta el estudio realizado, es difícil detectar un notable vencedor en cuanto a oportunidades y funcionalidades se refiere. Mientras una plataforma no hace énfasis en algún ámbito, otra lo aprovecha y lo mejora notablemente. De esta manera las cuatro plataformas han sabido mantenerse en el medio, y ser únicas en la industria. Las cuatro plataformas partieron de motivaciones similares, con el apoyo de grandes empresas en el fondo de todo movimiento. Incluso plataformas como Android, LiMo y Symbian

compartieron fundadores, siendo algo bastante importante que acotar pues se podría decir que las tres plataformas se conocieron desde el principio.

También han sabido jugar con sus funcionalidades. Hoy en día los Iphone se presentan como las alternativas más conocidas y usadas a nivel mundial, presentando inmensas aplicaciones de todo tipo al alcance de los usuarios y una interfaz sin igual. Apple ha sabido promocionar su producto y hacerlo llegar a aquellos que busquen esta tecnología. Sin embargo, les ha cedido paso a los usuarios que buscan otro tipo de características u otro tipo de terminales. OS X Mobile por ser una plataforma exclusiva de Apple no ha logrado llegar a expandirse totalmente en cuanto a dispositivos se refieren y en cuanto a aplicaciones ha logrado establecer muchas restricciones propias de la empresa. LiMo, Symbian y Android han aprovechado esto para formularse como tecnologías en constante desarrollo con una amplia gama de modelos de dispositivos móviles en el mercado. LiMo fue enfocada como un sistema que le da menos importancia a la interfaz gráfica, pero que actúa como middleware funcionalmente muy bien. Tiene planes futuros muy prometedores que le permitirán para el año 2010 ser una tecnología de código abierto totalmente abierta y al alcance de centenares de desarrolladores. Presenta soluciones y capacidad de desarrollo en lenguaje C/C++, un lenguaje bastante cercano al del sistema operativo, ofreciendo más expansión a nivel de desarrollo que lo colocan por encima de otras plataformas. Esta tecnología, aunque no cuenta con demasiada promoción en la actualidad, promete sin duda llegar a hacer mucho más grande en el futuro.

Por otro lado, Symbian también se ha hecho notar. Su alcance en el mercado no ha tenido límites y actualmente hay ya decenas de equipos a la venta que cuentan con esta tecnología. Presenta una adaptabilidad bastante buena, permite la programación en varios lenguajes que incluyen C++ y Java, así como el soporte de aplicaciones de tipo web. Su interfaz no es la mejor de las cuatro plataformas analizadas, pero sin duda es bastante amigable y fácil de usar. Igual que las demás plataformas ofrece muy buenas oportunidades a terceros para desarrollar nuevas aplicaciones de todo tipo y también está propensa a alistarse en el futuro a una gran tecnología con mucho que ofrecer a los usuarios.

Finalmente se encuentra Android, nuestra plataforma que fue objeto de estudio principal. Hoy en día, Google ha llevado a su tecnología a ganar importancia en el mercado. Cuenta actualmente con cientos de comunidades dedicadas a documentar el uso de la misma y las oportunidades a los terceros. Presenta una interfaz bastante comparable con la tan intuitiva interfaz del OS X Mobile, y actualmente cuenta con muchos proyectos a futuros así como gran cantidad de dispositivos móviles en el mercado.

En cuanto a aplicaciones que integren el posicionamiento global como principal atractivo hay muchas en casi todas las plataformas, porque en la actualidad representa una funcionalidad de gran demanda por los usuarios. Igual el disparo de alarmas se ha vuelto una característica casi por defecto en las plataformas, incluyendo este tipo de aplicaciones, desde su fabricación, en la mayoría de los dispositivos móviles disponibles. Aún falta mucho por crecer en todas las tecnología, pero ya muchas de ella cuentan con todas las herramientas necesarias para integrar esa tecnología, con el envío de mensajes de texto, correo electrónico, alarmas y demás. Android ya cuenta con las herramientas necesarias, plataforma en la que fue desarrollada la aplicación "Sistema de tareas y consulta de tráfico por posicionamiento global para dispositivos móviles bajo la plataforma Android de Google" por los autores del estudio comparativo presente. Definitivamente Android es una gran tecnología entre sus competidores y promete dominar la popularidad en el mercado. Para una completa documentación acerca del estudio realizado, los hechos, historia, funcionalidades y demás de cada plataforma, se recomienda la consulta del APÉNDICE G.

Funcionalidades/ Oportunidades	LiMo	Symbian	Os X Mobile	Android
Kernel	Basado en Linux	Symbian	OS X	Basado en Linux, con máquina virtual Dalvik
Conectividad	GSM, CDMA	3G, GSM, WiFi	3G, GSM, WiFi	3G, GSM, WiFi, GPRS
Interfaz intuitiva	Sí	No tan desarrollada	Sí	Sí
Instalación de nuevas aplicaciones	Sí	Compleja	Sí	Sí
Multitareas	----------	Sí	No	Sí
Ecosistema/Soporte multimedia	Sí	OVI	Itunes	Amazon
Bluetooth	Sí	Sí	Sí	Sí
SDK Disponible	Sí	Sí	Sí	Sí
Lenguajes de programación soportados	C/C++	C/C++, Java, Aplicaciones Web	Objective C	Java
Comunidades y soporte técnico	Aún en desarrollo	Sí	Sí	Sí
Dispositivos móviles disponibles en el mercado	Sí	Sí	Sí	Sí
Software completo	No	Sí	Sí	Sí
Emulador disponible para visualización de aplicaciones	Sí	Sí	Sí	Sí
Posibilidades de expansión futura	Sí	Sí	Sí	Sí

Tabla 10. Funcionalidades y oportunidades de desarrollo: LiMo, Symbian, OS X Mobile y Android

Fuente: Elaboración propia

CAPÍTULO V
RESULTADOS

Después de analizar el desarrollo relacionado con el presente proyecto y la aplicación que resultó como solución así como la información de respaldo que se recolectó en el camino, se pueden presentar los siguientes resultados:

- Se pudo finalmente lograr una evaluación y selección exitosa de las herramientas de desarrollo a utilizar durante todo el ciclo de vida. Después de evaluar sus funcionalidades y sus capacidades de integración, el Sistema de tareas y consulta de tráfico fue logrado finalmente con el uso del entorno de desarrollo Eclipse 3.3, con la ayuda de la plataforma seleccionada desde los inicios del proyecto: Android. Todos los sistemas implicados fueron de perfecto uso, logrando que mediante la metodología utilizada, se dieron con èxito todas sus fases logrando exitosamente la solución propuesta.
- Se desarrolló con éxito una aplicación que se valiera del dispositivo GPS del teléfono celular para configurar tareas que le permitieran al usuario contar con alarmas, recordatorios, envío de mensajes de texto y de correo electrónico, asociado a esas tareas. También se logró implantar un módulo de consulta de tráfico disponible para su interacción con el usuario.
- Siendo una aplicación basada principalmente en el posicionamiento global, se logró la integración de la misma con mapas perfectamente aptos y dotados de la información que el usuario necesitaba, gracias a la integración innata que poseía la plataforma seleccionada con Google Maps.
- Se desarrolló con éxito un módulo que se encargara del registro de coordenadas de los lugares de preferencia del usuario con cierta información asociada que le permitiera a este último diferenciarlos a su gusto.
- Se implantó y desarrolló un servidor de mensajes de texto que cuenta con la capacidad de manejar consultas por parte del usuario acerca de ciertas rutas mapeadas en la base de

datos. El servidor está capacitado para enviar mensajes de respuestas al usuario y de registrar los nuevos datos sobre las rutas que el usuario ofrezca en retroalimentación.

- Una vez desarrollado el servidor, se implantó con éxito una base de datos que manejará todos los datos referidos a la consulta de tráfico. Se pudo así almacenar información sobre las rutas mapeadas, tiempos en recorrer las rutas y demás información útil para el usuario.
- Luego de tener ya conocimientos bastante amplios acerca de la plataforma Android para dispositivos móviles, se logró estudiar el comportamiento de otras plataformas de la competencia: LiMo, Symbian y OS X Mobile (Iphone). Se realizó un estudio comparativo que permitió ver las ventajas y desventajas de todas las plataformas en cuanto a funcionalidad, historia y usabilidad por parte de los usuarios, así como una investigación de aplicaciones parecidas a la desarrollada en el presente proyecto.

CONCLUSIONES

Luego de haber pasado por todo el proceso de desarrollo anteriormente expuesto, se puede concluir en que el Sistema de tareas y consulta de tráfico por posicionamiento global basado en la plataforma Android de Google cumplió con sus objetivos y metas propuestas desde sus inicios. Se logró obtener una aplicación totalmente funcional que ofrece alternativas nuevas al usuario de este tipo de dispositivos móviles.

La aplicación ofrece varias funcionalidades al usuario que fueron basadas en la mayoría de las herramientas que ofrece Android como plataforma de código abierto. Igualmente se afianzaron los conocimientos acerca de la programación en el lenguaje Java, y se adquirieron otros. El sistema cuenta con algoritmos internos que fueron difíciles de construir, pero que finalmente lograron aprovechar al máximo las funcionalidades de Android que aún no habían sido integradas en una sola aplicación. Gracias a estudios previos elaborados para la propuesta del proyecto y el estudio comparativo que conformó los objetivos específicos, se logró concluir que la plataforma Android es una tecnología en crecimiento y es una de las que más oportunidades ofrece al usuario y a la comunidad de desarrolladores.

Como ejemplo, está la aplicación desarrollada, que logró integrar las capacidades del dispositivo GPS, alarmas de tiempo, alertas de proximidad, envío de mensajería de texto y correo electrónicos; haciendo uso de las capacidades en cuanto a interfaz gráfica de la plataforma Android.

Durante el desarrollo, se encontraron muchas dificultades. Por ser una plataforma de constante actualización, muchas funcionalidades fueron reemplazadas por otras mejores entre una versión y otra del SDK. Pero finalmente todas las dificultades fueron resueltas con éxito gracias a la gran cantidad de comunidades de desarrolladores de Android que existe en la actualidad, que hacen de esta tecnología una experiencia bastante placentera en lo que a desarrollo de software se refiere.

RECOMENDACIONES

Como recomendaciones acerca de la aplicación desarrollada, sus ambientes, funcionalidades e influencia en el usuario, se pueden citar las siguientes:

- La aplicación podría expandirse mucho más en cuanto a la funcionalidad. Se podrían fusionar tareas o investigar sobre un uso más amplio en el campo de la mensajería instantánea y los correos electrónicos.
- El sistema cuenta con una serie de rutas mapeadas de una manera predeterminada para cumplir con la consulta de tráfico que fue especificada como un objetivo. Sin embargo, se podría expandir esto a todas las rutas del área metropolitana de Caracas, Venezuela.
- Se podría implantar algún módulo extra que presente estadísticas en tiempo real, independientemente si el usuario consulta el tráfico o no.
- En el desarrollo de la presente solución, se encontraron dificultades debido a las constantes actualización del SDK de Android. Ante esto, se recomienda documentarse al máximo acerca de las actualizaciones del mismo y los cambios realizados entre una y otra, para que el desarrollo no se vea interrumpido por una modificación del código desarrollado de aplicaciones basadas en esta plataforma.
- En base a cualquier algoritmo, se recomienda también la completa documentación acerca de todos los caminos que se pueden tomar. Android ofrece un sinfín de librerías y funcionalidades que pueden ser implementadas para cualquier algoritmo. Se recomienda evaluar y estudiar las opciones y oportunidades que ofrece Android, antes de llevarlos al código.
- Se recomienda también una posible expansión de la aplicación en cuanto a un perfeccionamiento de la interfaz de la aplicación evaluando posibles mejoras.
- Se recomienda esta plataforma para futuros trabajos por su fácil acceso, programación, integración y su ya comprobada infinidad de oportunidades que ofrece al desarrollador con todas sus herramientas.

REFERENCIAS BIBLIOGRÁFICAS

- Google Android. http://code.google.com/intl/es-ES/android/what-is-android.html
- Steve McConnell (1997). "Desarrollo y gestión de Proyectos Informáticos". Editorial McGraw Hill.
- Reto Meier (2008). "Professional Android Application Development". Wrox Programmer to Programmer.
- Open Handset Alliance. http://www.openhandsetalliance.com/ android_overview.html
- Google's T-Mobile G1 Android Phone: The First Review. http://technology.timesonline.co.uk/tol/news/tech_and_web/article4810954.ece
- Mobile Review's Android First Look. http://www.mobile-review.com/soft/2007/android-en.shtml
- Greg Kumparak (2008). "Android vs. LiMo: What's the difference?" http://www.mobilecrunch.com/2008/05/14/android-vs-limo-whats-the-difference/
- Kote (2008). "Android vs LiMo". http://www.xatakamovil.com/sistemas-operativos/android-vs-limo
- Stuart Corner (2008). "Watch out for the LiMo, Android!" http://www.itwire.com/content/view/17407/127/1/1/
- Sin autor. http://es.wikipedia.org/wiki/LiMo_Foundation
- LiMo Foundation. http://www.limofoundation.org/
- Sin autor. http://en.wikipedia.org/wiki/LiMo_Foundation
- Rob Jackson (2008). "The Android vs. LiMo Conflict" http://phandroid.com/2008/05/14/android-limo-conflict/
- Sin autor (2008). http://www.gomonews.com/android-vs-limo/
- Sin autor (2009). http://gruaboy.blogsome.com/2009/04/24/android-vs-symbian/
- Antone Gonsalves (2008). "Android, Symbian expectec to become one .OS" http://www.informationweek.com/news/mobility/business/showArticle.jhtml?articleID=209600592

- Martyn Williams (2007). "Android just another Linux platform, says Symbian CEO" http://www.networkworld.com/news/2007/110607-android-just-another-linux-platform.html
- Sin autor. http://es.wikipedia.org/wiki/Symbian_OS
- Symbian Foundation. http://www.symbian.org/
- José Elías (2008). "Iphone vs Android, la batalla se avecina..." http://eliax.com/index.php?/archives/4563-iPhone-vs-Android,-la-batalla-se-avecina,...html
- Gina Trapani (2009). "Android Versus Iphone 3.0: The Showdown" http://lifehacker.com/5173441/android-versus-iphone-30-the-showdown
- Ed Hardy (2008) "Iphone vs. Android: a Study in Contrasts" http://www.brighthand.com/default.asp?newsID=14142
- Sin autor. http://es.wikipedia.org/wiki/IPhone_OS
- Sin autor (2009). "Comparativa entre sistemas operativos móviles" http://diarioandroid.com/2009/05/06/comparativa-entre-sistemas-operativos-moviles/#more-1717
- Aspicore. http://www.aspicore.com/en/tuotteet_tracker.asp?tab=2&sub=2
- MGMaps News. http://www.mgmaps.com/news.php?item=145
- Android Developer Community. http://developer.android.com/community/index.html
- Davanum Srinivas. "Show the code!". http://davanum.wordpress.com/2007/12/22/android-send-email-via-gmail-actually-via-smtp/
- Webside Baker. "Servidor SMS". http://servidorsms.goldeye.info/websitebaker/pages/bienvenidos.php
- Bruce P. Blackshaw. "Returning a JDBC result set from an Oracle stored procedure". http://www.enterprisedt.com/publications/oracle/result_set.html