Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Aplicación para la gestión de perfiles en sistemas operativos Android

Autor

David Valdés Fernández

Tutores

Ing. José Gustavo Suárez Matilla

Ing. Miriela Velázquez Arias

La Habana, junio de 2016

"Tus clientes más descontentos son tu mayor fuente de aprendizaje."

Bill Gates

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a que haga el uso que estime pertinente con el mismo.

Para que así conste se firma el presente a los ___ días del mes de _____ del año 2016.

Autor

_____

David Valdés Fernández

Tutores

_____ _____

Ing. Miriela Velázquez Arias Ing. José Gustavo Suárez Matilla

Dedicatoria

A mis abuelos, que siempre van a estar presente en todo momento en mi vida, que forjaron con sus experiencias, conocimientos y amor, el hombre culto, educado y familiar que soy hoy.

A mi madre, a mi padre y al gordo, por el sacrificio incondicional y por apoyarme en todas mis decisiones, por ayudarme a levantarme en todos los momentos difíciles.

Agradecimientos

A mi madre, son tantas cosas por las que agradecerte que no las puedo resumir en unas oraciones, pero ten claro, que todo lo que soy hoy es gracias ti.

A mi padre, no hay distancia que afecte nuestra relación, el apoyo de tu parte siempre ha sido incondicional, al igual que mamá no puedo, gracias.

Amado han sido muchas mis malcriadeces y aún así me distes más de lo que estuvo a tu alcance, nos has querido y tratado como lo que soy para ti, un hijo.

A mi hermano Daniel, me llevaste por muchos caminos, aprendí muchas cosas de ti, convivimos juntos largos años de universidad y muchas veces me tirabas de la cama, gracias por todo.

A Sussy, mi reina, mi amor, muchas fueron nuestras discusiones por mi estrés, pero nada de eso te detuvo, me ayudaste siempre, me alentaste cuando pensé que no tenía tiempo, me distes los empujones que necesité para seguir.

Adrián, que decir de ti, mi amigo, mi hermano, mi compañero, mi subordinado, esta universidad hizo de nosotros dos lo que hoy somos, tu sacrificio, las noches sin dormir, todo, a ti, te lo agradezco todo.

A Angelito y Wilfredo, la sala es mi habitación y el teléfono es mío, estoy seguro que no van a tener un gitano como yo.

Resumen

El uso de las tecnologías móviles ha incrementado significativamente. Cada día surgen nuevos usuarios y aumentan las funcionalidades que ellos necesitan. En estos dispositivos se instalan varias aplicaciones para abarcar en su mayoría estas funcionalidades, consumiendo en gran medida los recursos del dispositivo.

En este trabajo se realizó el diseño e implementación de la herramienta GesPer, aplicación encargada de gestionar perfiles de configuración en dispositivos con sistema operativo *Android*. La principal función de GesPer es automatizar tareas a través de perfiles, agrupando un conjunto de las principales funciones y reduciendo de esta manera el consumo de recursos del equipo en el que se instala.

Como resultado de la investigación y desarrollo se obtuvo un producto que realiza configuraciones en el dispositivo sobre los servicios de *wi-fi,* audio, y muestra aplicaciones seleccionadas para cada perfil en el rango de fechas definidas. Esto permite a los futuros usuarios comodidad y ahorro de tiempo en su vida cotidiana. Mediante las pruebas de validación realizadas sobre la aplicación se demostró que el consumo de memoria RAM, CPU y de batería en el dispositivo móvil es menor.

Palabras clave: perfil de configuración, servicios, automatizar, Android.

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

En la actualidad resulta evidente el significativo auge de las Tecnologías de la Información y las Comunicaciones (TIC). Específicamente la tecnología móvil destaca por los beneficios y usos que ofrece. Según un estudio de *Mobile Behaviour Report*, esta tecnología resulta cada día más imprescindible para la sociedad. Dicho estudio reveló que el 85% de los usuarios consideran a los dispositivos móviles como una herramienta esencial en su cotidianidad; un porcentaje que se eleva al 90% en el caso de los jóvenes entre 18 y 24 años (1).

Paulatinamente los dispositivos móviles han evolucionado, ya no solo se limita al envío y recepción de llamadas y mensajes de texto. Han incorporado funciones como Asistentes Digitales Personales (PDA[1]), cámara, agenda electrónica, alarma, calculadora, radio portátil, GPS[2] y reproductor multimedia. A este tipo de evolución del teléfono móvil se le conoce como teléfono inteligente (*smartphone*). De manera general, es posible realizar una gran variedad de acciones mediante un dispositivo pequeño y portátil.

Las tecnologías antes expuestas requieren de un sistema operativo (SO) para su funcionamiento e interacción con los usuarios. Entre los SO vinculados a la tecnología móvil destacan *Android*, *IOs* y *Windows Phone.* En un estudio realizado por *International Data Corporation* (IDC) en agosto de 2015, como parte del continuo análisis del mercado asociado a las tecnologías móviles, se concluyó que un 82,8% de usuarios hace uso de *Android*, 13,9% utilizan iOs, un 2,6% usan *Windows Phone* y un 0,7% utilizan otro sistema operativo (2).

A partir de la creciente dependencia sociedad-dispositivo_móvil, los desarrolladores de aplicaciones para SO *Android* crean soluciones dirigidas a proporcionar mayor comodidad explotando al máximo estos dispositivos. Por lo que teniendo en cuenta elementos como la usabilidad, organización y gestión de configuraciones personalizadas, se desarrollan aplicaciones para la administración y automatización de tareas, a través de las cuales es posible crear flujos de actividades programados para facilitar y brindar ayuda al usuario. Estas aplicaciones permiten crear secuencias de eventos iniciadas por acciones específicas.

Por otro lado, existen soluciones para la gestión del comportamiento de las aplicaciones, dígase la activación o desactivación de los servicios asociados a las mismas. También están aquellas que permiten, a través de

[1] Asistente Digital Personal: Manejan funciones de gestión de información personal (38).
[2] *Global Positioning System* - Sistema de Posicionamiento Global (45)

tareas programadas, habilitar y deshabilitar servicios como *wi-fi*, *bluetooth* o el acceso a datos. Además, es común encontrar las que permiten personalizar las configuraciones de los tonos del dispositivo a través de perfiles de configuración[3]. Dado que estas aplicaciones están dirigidas, en su mayoría, a satisfacer una necesidad particular, es común que los usuarios deban instalar varias de ellas para cubrir total o parcialmente la gestión de los servicios, aplicaciones y tareas del dispositivo. Esto trae consigo los siguientes inconvenientes:

- Conflicto entre aplicaciones: cada una posee sus propios requerimientos y es posible que un conflicto entre ellas pueda afectar el funcionamiento de otras aplicaciones.
- Alto consumo de recursos: por la cantidad de servicios corriendo en segundo plano a la vez, incidiendo directamente en la RAM[4], CPU[5] y duración de la batería. Este último es uno de los aspectos que más llama la atención de los desarrolladores de aplicaciones *Android*.
- Usabilidad: se requieren de varios pasos e interacción con diversas aplicaciones para gestionar tantos aspectos a la vez.

A partir de la actual revolución sobre las tecnologías, la Universidad de las Ciencias Informáticas (UCI) no queda exenta del desarrollo de aplicaciones en esta esfera, por lo que surge el Laboratorio de *Android* (DroidLab) para el desarrollo de aplicaciones para este sistema operativo. En busca de oportunidades de negocio en el Caribe para el desarrollo de aplicaciones móviles, el laboratorio DroidLab está optando por insertarse como laboratorio para desarrollo de aplicaciones móviles y capacitación para profesionales, en este caso en Android (*mHubs)* en el proyecto *Caribbean Mobile Innovation Project* (CMIP) que lleva a cabo los Bancos Mundiales y el gobierno de Canadá.

Teniendo en cuenta el auge del mercado de aplicaciones *Android*; la necesidad de gestionar los servicios, aplicaciones y tareas de manera que se consuma menos recursos y se disminuya el tiempo necesario por los usuarios para gestionar una configuración del perfil; así como la identificación de posibles oportunidades de negocio, se define como **problema a resolver:** ¿Cómo facilitar la gestión de servicios, aplicaciones y tareas de los dispositivos móviles a través de perfiles de configuración de forma tal que se minimice el consumo de recursos del sistema?

[3] Grupo de rasgos característicos (39).
[4] Memoria de Acceso Aleatorio: Dispositivo donde se almacenan datos e instrucciones (40).
[5] Unidad Central de Procesamiento: Es el *hardware* dentro de un dispositivo programable que interpreta las instrucciones de un programa informático (41).

Conociendo el problema definido con anterioridad se determina como **objeto de estudio** las soluciones que gestionan los servicios, aplicaciones y tareas de los dispositivos móviles. Enmarcando como **campo de acción** las herramientas que gestionan los servicios, aplicaciones y tareas de los dispositivos móviles de sistema operativo *Android*.

Como **objetivo general** se plantea desarrollar una herramienta que gestione los servicios, aplicaciones y tareas de dispositivos móviles con SO *Android*, a través de perfiles de configuración minimizando el consumo de los recursos batería, memoria RAM y CPU.

Para dar cumplimiento al objetivo general se definen como **objetivos específicos:**

1. Definir el marco teórico de la investigación.
2. Analizar metodologías, herramientas y tecnologías que contribuyan al desarrollo de la aplicación.
3. Diseñar la propuesta de solución.
4. Implementar la solución a partir de los requisitos funcionales y no funcionales definidos.
5. Validar la propuesta de solución mediante pruebas.

Se propone como **idea a defender** que con el desarrollo de una aplicación *Android* que posibilite gestionar las tareas, aplicaciones y servicios de los dispositivos móviles a través de perfiles de configuración, disminuirá el consumo de recursos del mismo, mejorando el tiempo de duración de la batería, la velocidad del procesador y la memoria RAM.

Con el propósito de dar cumplimiento al objetivo general antes propuesto, se plantean las siguientes **tareas de la investigación:**

1. Definición de los principales conceptos asociados al sistema operativo y al desarrollo de aplicaciones *Android*.
2. Definición de la arquitectura del sistema.
3. Definición de los requisitos funcionales y no funcionales a implementar.
4. Diseño e implementación de las clases y métodos que den solución a los requisitos definidos.
5. Ejecución de casos de pruebas de aceptación y experimentales.

Los **métodos teóricos** empleados fueron:

- **Analítico-Sintético:** Se hace uso de este método para enmarcar los elementos teóricos de la investigación facilitando el análisis y comprensión de la documentación consultada, contribuyendo a realizar comparaciones entre disímiles aplicaciones *Android*, extraer y precisar los elementos y características fundamentales del diseño de la propuesta de solución y establecer las conclusiones de la investigación.
- **Histórico-Lógico:** Se utiliza para realizar un estudio del estado del arte, del desarrollo de las aplicaciones *Android* que automatizan tareas.

Los **métodos empíricos** empleados fueron:

- **Observación:** Se emplea como registro visual de lo que ocurre en una situación real, calificando los acontecimientos de acuerdo con esquemas previstos. Se utiliza para analizar el comportamiento de las aplicaciones estudiadas.
- **Experimental:** Se emplea durante el proceso de pruebas experimentales para comprobar el cumplimiento del objetivo general planteado.

El trabajo de diploma está estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica.

En este capítulo se realiza un estudio del estado del arte de aplicaciones *Android* para la gestión de servicios, tareas y aplicaciones, se abordan elementos teóricos vinculados a la investigación, y se definen tecnologías y herramientas a utilizar, así como la metodología de desarrollo que guíe la implementación de la aplicación.

Capítulo 2: Análisis y diseño de la solución.

En este capítulo se realiza el modelado de los procesos de negocio y posteriormente se definen los requisitos funcionales y no funcionales. Se muestran, además, los resultados de la aplicación de métricas para la validación de requisitos. Se describe el diseño de la solución y se especifican las características y patrones de diseño utilizados.

Capítulo 3: Implementación y prueba de la solución.

En este capítulo se describen los principales aspectos o artefactos vinculados a la implementación. Se especifican las pruebas ejecutadas a la aplicación con el objetivo de verificar su correspondencia con los requerimientos definidos.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el presente capítulo se define el marco teórico de la investigación. Se realiza un análisis del estado del arte de las principales aplicaciones del sistema operativo *Android* que gestionan los servicios, aplicaciones y tareas. Se analizan tecnologías y herramientas que apoyen el diseño y la implementación de la aplicación, así como la metodología de desarrollo de *software* a utilizar.

1.2 Marco conceptual

A continuación, se definen algunos conceptos asociados a la investigación.

***Android*:** Es un sistema operativo orientado a dispositivos móviles, basado en una versión modificada del núcleo Linux. *Android* es un sistema abierto y multitareas que permite a los desarrolladores acceder a las funcionalidades principales del dispositivo mediante aplicaciones (3).

Automatizar: Sistema en que se transfieren tareas, realizadas habitualmente por operadores humanos a un conjunto de elementos, con el objetivo de mejorar la productividad reduciendo el costo, mejorar las condiciones de trabajo y la disponibilidad de los servicios (4).

Permisos de administración*:* En sistemas operativos de tipo Unix (Sistema Operativo Portátil, Multitarea y Multiusuario), es el término empleado para definir a la cuenta de usuario que posee todos los derechos para modificar lo que ocurre en el sistema (5).

Perfil: Un perfil es el conjunto de información que contiene su configuración, preferencias, mensajes de correo, contraseñas, libretas de direcciones y certificados. Los programas de correo utilizan los perfiles para organizar la información de distintos usuarios (6).

Perfil de configuración: El perfil de configuración es una colección de opciones que permiten que el dispositivo tenga el aspecto y funcione de la manera que se desee. Contiene la configuración para fondos de escritorio, protectores de pantalla, preferencias de puntero, configuración de sonido y otras características (7).

Disparador: Función que se ejecuta cuando se produce una operación en las bases de datos. Programas a los que se les asigna un nombre de objeto, se asocia con una tabla determinada, y se activa cuando ocurre un evento en la tabla, como una inserción, actualización o eliminación (8).

***Software*:** Suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo (9).

***Software* libre:** Es el *software* que respeta la libertad de los usuarios y la comunidad. Los usuarios tienen la libertad de copiar, ejecutar, distribuir, estudiar, modificar y mejorar el *software* (9).

Código fuente: Texto escrito en un lenguaje de programación. Debe traducirse a lenguaje de máquina para que pueda ser ejecutado por la computadora (10).

***Open source*:** Tener acceso al código fuente, *software* distribuido y desarrollado libremente, tiene como punto de vista los beneficios prácticos de compartir el código (11).

APK: Los archivos de extensión apk, son utilizados para instalar *software* en el sistema operativo *Android* (12).

1.3 Aplicaciones de automatización de acciones en dispositivos móviles

De forma general todas las aplicaciones que automatizan acciones usan una acción desencadenante para iniciar la ejecución de un comportamiento determinado del dispositivo. Difieren en que algunas trabajan sus secuencias en forma de flujos, otros en marcos, entre otros. En estos flujos se automatizan procesos quedando anclados a bloques que determinan toda clase de acciones (13). Los marcos son la forma en que los dispositivos con esta aplicación instalada reaccionan como respuesta ante una aplicación ejecutada o ante el mismo uso del dispositivo (14).

1.3.1 Análisis de aplicaciones similares

Durante el proceso de investigación no se encontró una aplicación que cubra todos los aspectos que se desean gestionar; o sea que permita gestionar los servicios, aplicaciones y tareas del dispositivo. Como alternativa se decide analizar aplicaciones que gestionen estos aspectos de forma independiente.

1.3.1.1 Aplicaciones para la planificación de tareas

- ***Trigger***

Un *trigger* o disparador, en una base de datos, es un procedimiento que se ejecuta de manera automática cuando se cumple una condición establecida al realizar una operación. Dependiendo de la base de datos, los *triggers* pueden ser de inserción (*INSERT*), actualización (*UPDATE*) o eliminación (*DELETE*). Algunas bases de datos pueden ejecutar *triggers* al crear, borrar o editar usuarios, tablas, bases de datos u otros objetos, lo que facilita la administración de la base de datos. Además, pueden generar valores de columnas, prevenir errores de datos, sincronizar tablas, modificar valores de una vista, entre otras funciones (8).

Trigger es una aplicación *Android* que automatiza acciones dependiendo de los disparadores que se configuren. Con esta aplicación se pueden realizar varias tareas, como que el reproductor de audio se inicie al activar el *bluetooth* y se conecte a la lista de descarga automáticamente; también se puede configurar el perfil de usuario a modo silencio o cualquier otro una vez que se conecte a una *wi-fi* (13). Trigger es una aplicación muy útil, pero no todos los usuarios pueden hacer uso de sus funcionalidades, algunas de estas para su funcionamiento requieren de permisos de administración para poder ser ejecutadas en el dispositivo en el que se encuentra instalada.

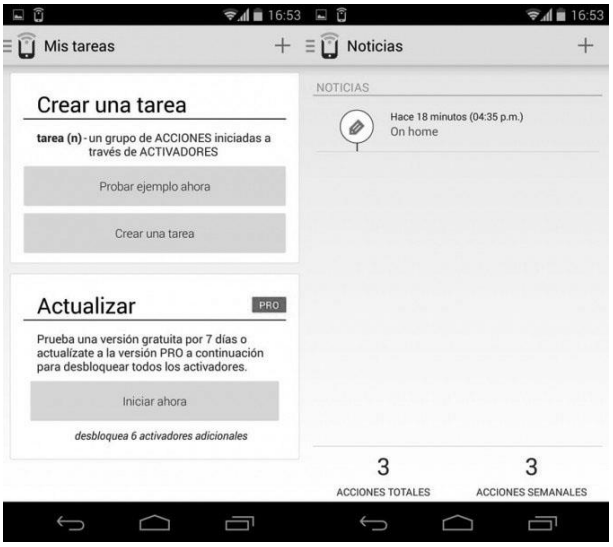


Figura 1: Funcionalidad Crear tarea de la aplicación Trigger.

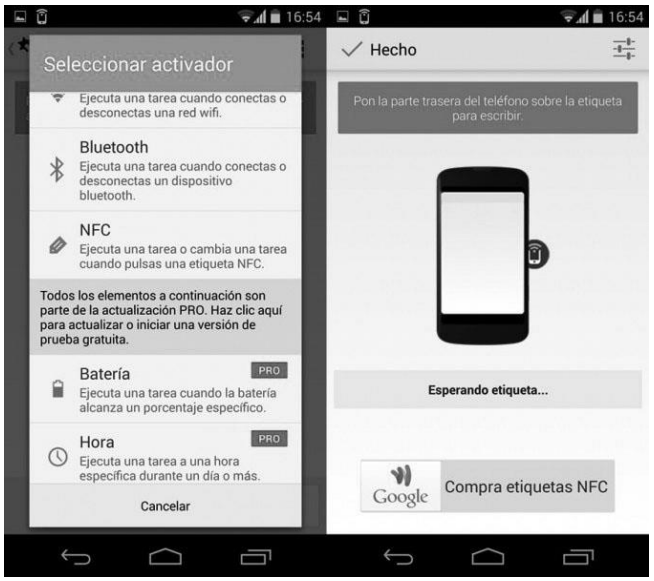


Figura 2: Funcionalidad Seleccionar activador de la aplicación Trigger.

- ***Tasker***

Es una aplicación para la automatización de tareas en dispositivos móviles con sistema operativo *Android*. *Es* completamente administrativa, necesita para su ejecución que el dispositivo posea todos los permisos de administración. Ofrece soluciones más completas sin aumentar enormemente la complejidad en el proceso. Con *Tasker* se pueden crear tareas tanto sencillas como complejas, e incluso se puede llegar a ejecutar código fuente desde la consola de Android en forma de *scripts*[6] (13)

Algunas de las operaciones que se pueden realizar con *Tasker* son las siguientes (15):

- Código de bloqueo de aplicaciones sensibles.
- Cambiar la configuración del teléfono.
    - Aplicación: según la aplicación que se esté ejecutando puede modificarse el tiempo de bloqueo de la pantalla.
    - Tiempo: de acuerdo a la hora del día bajar o subir el brillo de la pantalla.
    - Ubicación: cambiar el timbre en la oficina a modo silencioso o vibrador.

[6] Conjunto de instrucciones almacenadas en un archivo de texto, que deben ser interpretados línea a línea en tiempo real para su ejecución (44).

- Despertador: seleccionar una canción al azar de la colección de música del dispositivo.
- Ejecutar el reproductor de audio una vez colocada la tarjeta SD.
- De acuerdo al ángulo de inclinación del dispositivo con respecto al suelo activar o desactivar el altavoz durante una llamada.
- Enviar un mensaje de emergencia con la ubicación por GPS.
- Reasignar botones.

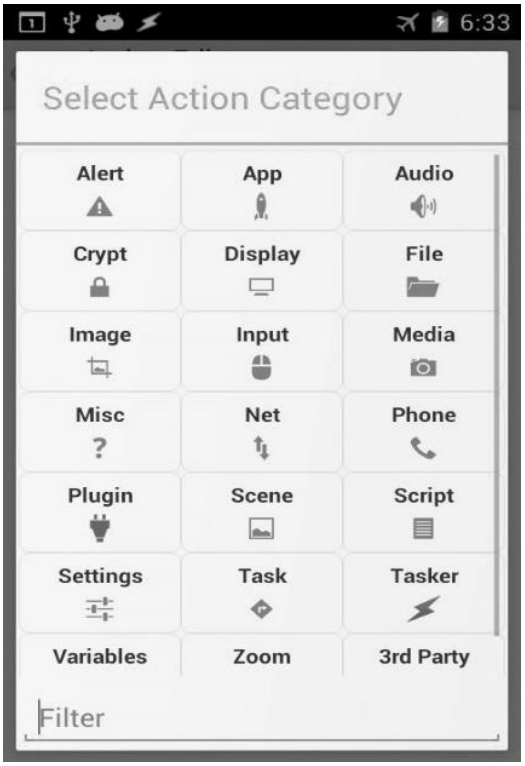


Figura 3: Funcionalidad Selección de categorías de acción de la aplicación Tasker.

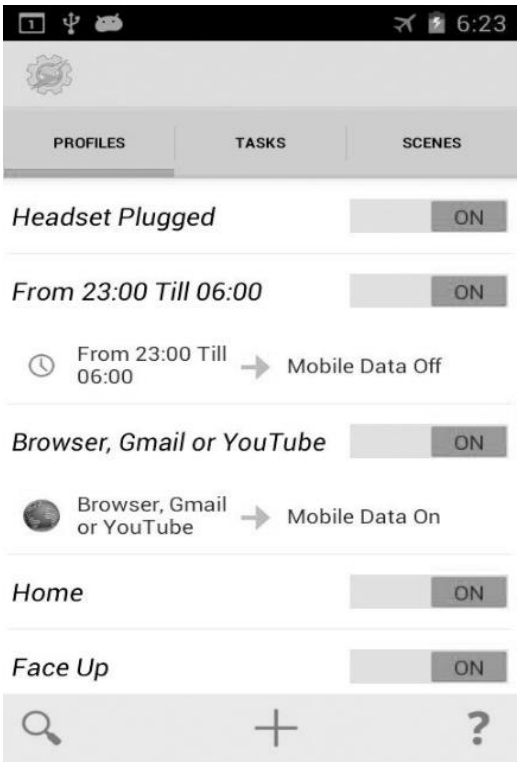


Figura 4: Funcionalidad Perfiles de la aplicación Tasker.

1.3.1.2 Aplicación para la planificación de flujos de tareas

- ***Automate***

Esta aplicación permite definir flujos de tareas en un estilo de pseudocódigo[7] logrando un alto nivel de complejidad y detalles. Un ejemplo de las diversas funcionalidades que ofrece *Automate* es la detección de ubicación, según el lugar donde se encuentre activada o desactivada la *wi-fi* (13).

Automate posee un conjunto de 170 bloques que pueden ser usados para crear todo tipo de tareas. Entre estos bloques los que más se destacan son (16):

- Modo avión.

[7] Lenguaje de programación algorítmico, lenguaje intermedio entre el lenguaje natural y cualquier lenguaje de programación específico (42).

- Inicio o cierre de aplicaciones.
- Volumen de audio.
- Llamadas.
- Eventos de calendario.
- Micrófono silenciado.
- Notificación.
- Grabación de audio.
- Pantalla: brillo, orientación.
- Fondos de pantalla.
- SMS: componer, enviar, enviado, recibido.

Esta aplicación resulta de gran utilidad, actualmente se encuentra en una versión beta en la tienda de Google, por lo que para su consumo requiere de un pequeño costo de descarga. Posee muchas ventajas y funcionalidades, pero su uso resulta muy complejo para usuarios con pocos conocimientos de programación, imposibilitándoles realizar grandes automatizaciones con la misma.

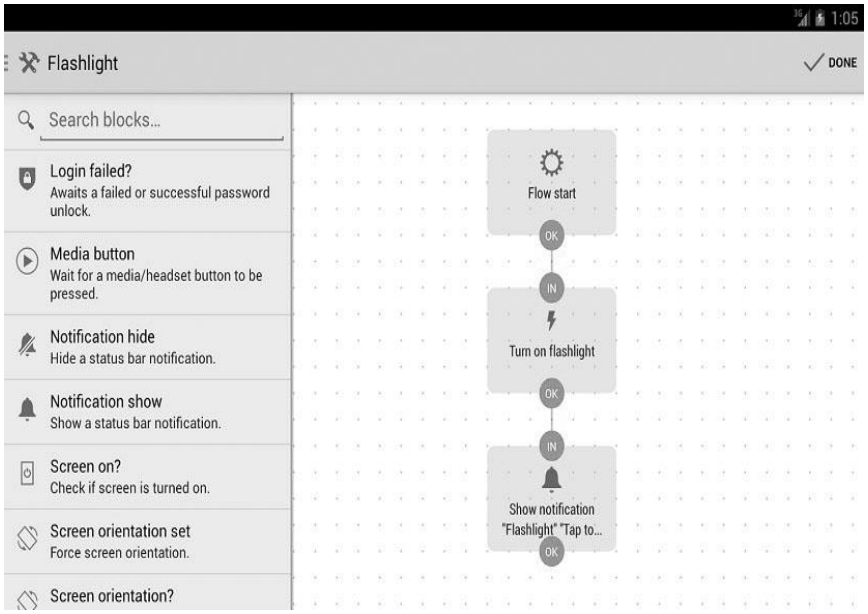


Figura 5: Funcionalidad Crear flujo de la aplicación Automate.

1.3.1.3 Aplicaciones para la gestión y planificación de servicios

- **MacroDroid**

Se puede utilizar para desactivar el *bluetooth*, silenciar el teléfono, desactivar los datos cuando se está en un lugar determinado. Se puede configurar para leer los mensajes de texto en voz alta o hacer una carga rápida de fotos a las redes sociales (17).

Lo que básicamente permite MacroDroid es crear marcos, de forma que el dispositivo móvil reaccionará de manera automática a ciertos estímulos; creando respuestas para cuando se agote la batería del dispositivo, cuando se tome una fotografía o al conectarse a una red (14).

Figura 6: Funcionalidad Adicionar acción de la aplicación MacroDroid.

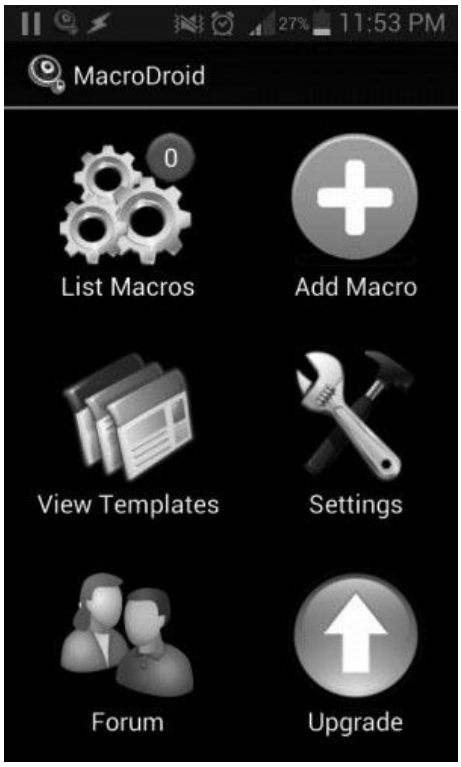


Figura 7: Interfaz de la aplicación MacroDroid.

- ***Wi-Fi Matic***

Con esta aplicación se puede programar la conexión o desconexión automáticamente a una red *wi-fi* según el lugar en que se encuentre (18). *Wi-Fi Matic* utiliza los identificadores de las celdas de telefonía móvil, para conocer la ubicación relativa del terminal y de las redes *wi-fi*, por lo que no necesita tener activos los servicios de ubicación por GPS. Esta aplicación activa la *wi-fi* al acercarse a una red y la desactiva al alejarse. Una vez memorizados los lugares, *Wi-Fi Matic* desconectará la conexión *wi-fi* del dispositivo al alejarse para ahorrar batería, conectando de nuevo la red y la conexión al punto de acceso memorizado al acercarse al rango de alcance (19).

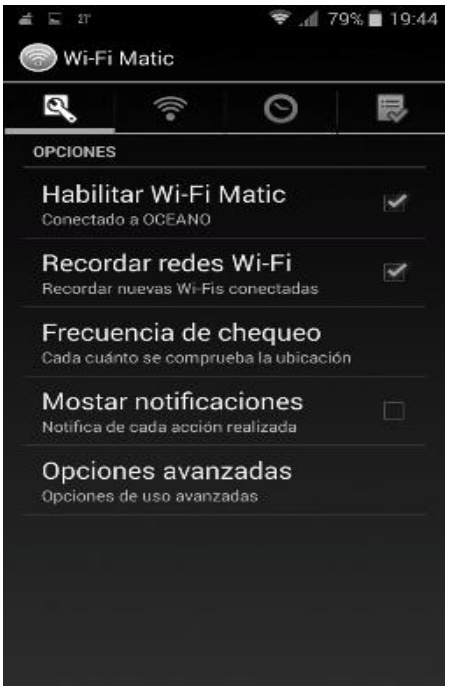


Figura 8: Funcionalidad Opciones de la aplicación Wi-fi Matic.

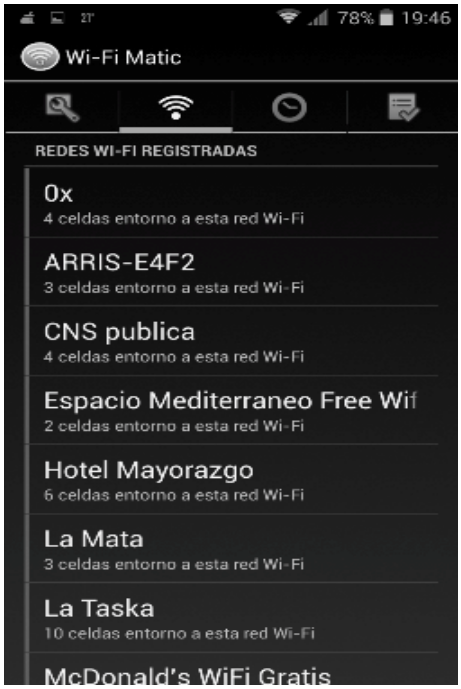


Figura 9: Lista de redes wi-fi de la aplicación Wi-fi Matic.

1.3.1.4 Herramientas para la gestión de aplicaciones

- ***Gemini app manager***

Permite realizar copias de seguridad a las aplicaciones instaladas; moverlas desde la tarjeta interna a la externa del dispositivo y viceversa; desinstalarlas y desactivarlas, detener los procesos en ejecución; borrar caché de las mismas y controlar los permisos de cada herramienta. Esta aplicación realiza estas operaciones por lotes, lo que permite ahorrar tiempo (20).

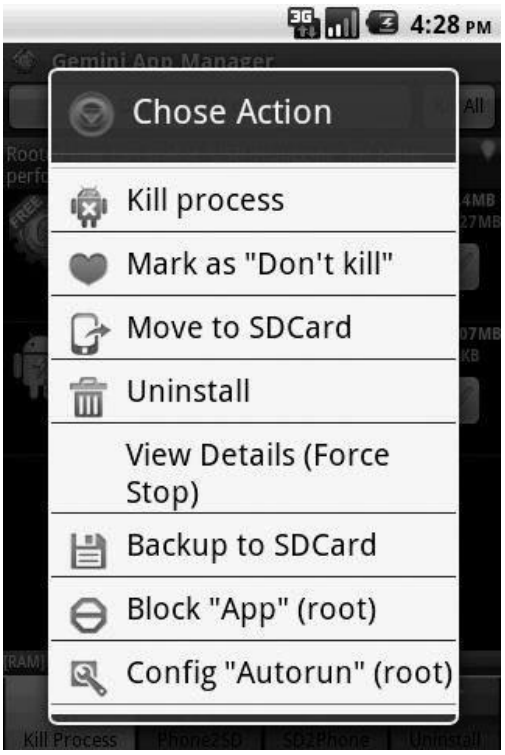


Figura 10: Funcionalidad Acciones del sistema de la aplicación Gemini app manager.

- ***Android App Manager***

Esta aplicación brinda la posibilidad de realizar búsquedas rápidas de aplicaciones por nombres; fecha de instalación; tamaño y demás; realizar copias de seguridad y desinstalación por lotes; controlar las actualizaciones de los programas; brinda información del uso de cada aplicación; de este modo permite conocer sobre aplicaciones que se utilizan poco en el dispositivo (20).

Figura 11: Lista de Procesos en Ejecución de la aplicación Android App Manager.

1.4 Metodología de desarrollo de *software*

Según Somerville, para muchas personas el *software* es solo un programa de computadora. Sin embargo, comenta que son todos aquellos documentos asociados a la configuración de datos que se necesitan para hacer que estos programas operen de manera adecuada. Estos productos de *software* se desarrollan para algún cliente en particular o para un mercado en general. Para el diseño y desarrollo de proyectos de *software* se aplican metodologías, modelos y técnicas que permiten resolver el problema (21).

Metodologías ágiles

Existen diversas metodologías ágiles. Todas pretenden de una forma u otra desarrollar *software* de manera rápida y respondiendo satisfactoriamente a los cambios que surjan durante el ciclo de vida del proyecto. El Manifiesto Ágil que se desarrolló en EEUU en el año 2011, plantea los siguientes principios (22):

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de *software* que le aporten valor.
- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- La atención continua a la calidad técnica y el buen diseño mejora la agilidad. La simplicidad es esencial.
- En intervalos regulares el equipo reflexiona respecto a cómo llegar a ser más efectivos y según esto ajusta su comportamiento.

Metodología de desarrollo de *software* AUP

La metodología de desarrollo de *software* AUP (*Agil Unified Process*) es una versión simplificada de la metodología de desarrollo RUP (*Rational Unified Process*). Describe de manera fácil y simple el desarrollo de *software* de negocio, empleando técnicas ágiles y conceptos que se mantienen válidos en RUP. Aplica técnicas como:

- Desarrollo dirigido por pruebas.
- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.

AUP se preocupa especialmente por la gestión de riesgos. Propone que los elementos con alto riesgo obtengan prioridad en el desarrollo y sean abordados en etapas tempranas del mismo.

Establece cuatro faces que transcurren de manera consecutiva:

1. Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo_de_desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

4. Transición: El sistema se lleva a entornos donde se somete a pruebas de validación y aceptación; finalmente se despliega en los sistemas de producción.

Metodología de desarrollo de *software* AUP versión UCI

La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP, con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la misma. La versión de la metodología desarrollada por la universidad decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma y se unifican las restantes fases de la metodología de desarrollo de *software* AUP en una sola, nombrada Ejecución y agregándose también una nueva fase llamada Cierre (23).

Tabla 1: Fase variación AUP-UCI.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases(Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración Construcción Transición	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como la ejecución y se realizan las actividades formales de cierre del proyecto.

Para el apoyo de la implementación de la solución se empleará la metodología de desarrollo de *software* AUP modificada por la UCI, puesto que es la metodología definida por el centro DroidLab para el desarrollo de proyectos.

1.5 Herramientas y tecnologías a utilizar

A continuación, se definen las herramientas y tecnologías que apoyarán el desarrollo de la aplicación.

- ***Android Studio* 1.4**

Software creado directamente por *Google* para el desarrollo de aplicaciones móviles basado en IntelliJ IDEA. *Android Studio* agrega elementos específicos para *Android* que *Google* ha incluido. Tiene herramientas especializadas que hacen el proceso más fácil, rápido y sencillo. Utiliza *Gradle* como compilador. Posee un editor gráfico que permite generar interfaces sin la necesidad de crear código. Se tiene acceso a herramientas personalizadas y optimizadas para trabajar en *Android*, de forma que las interfaces son específicamente pensadas para el trabajo en entornos *Android*. Posee completamiento automático de código y corrección de errores. Es una experiencia 100% *Android* (24).

- ***Gradle* 2.5**

Es un sistema de construcción avanzado que permite la creación de una lógica de construcción personalizada a través de *plugins*. Algunas de las características que hicieron elegir *Gradle* al equipo de *Android* son las siguientes (25):

- Lenguaje específico del dominio (DSL) para describir y manipular la lógica de construcción.
- Construye archivos basados en Groovy y permite la mezcla de elementos declarativos a través de la DSL, y la utilización de código para manipular los elementos de la DSL para proporcionar lógica personalizada.
- Incorpora gestor de dependencias a través de Maven y/o Ivy.
- Flexible. Permite utilizar mejores prácticas, pero no fuerza su propia manera de hacer las cosas.

- Los *plugins* pueden exponer su propio DSL y su propia API[8] creando archivos para su uso.
- Estampación de la API permitiendo integración con el IDE.

Permite generar configuraciones personalizadas y compilar de forma estable, obteniendo gran control sobre el producto final que se está compilando.

- **Maven**

Maven se utiliza en la gestión y construcción de *software*. Posee la capacidad de realizar tareas claramente definidas, como la compilación del código y su empaquetado. Es decir, hace posible la creación de *software* con dependencias incluidas dentro de la estructura del JAR. Es necesario definir todas las dependencias del proyecto (librerías externas utilizadas) en un fichero propio de todo proyecto Maven, el POM (*Project Object Model*). Este es un archivo en formato XML que contiene todo lo necesario para que cuando se quiera generar el fichero ejecutable de la aplicación, contenga todo lo que necesita para la ejecución (26).

- **Java**

Java es un lenguaje de programación con elementos de C y C++. Es un lenguaje de propósito general, de alto nivel, y orientado a objeto. El lenguaje Java es a la vez compilado e interpretado. Es un lenguaje dinámico, debido a que las clases son cargadas en el momento en que son necesitadas. Posee varias capas de seguridad para evitar que programas maliciosos pudiesen causar daños (27).

- **Git 2.6.3**

Es un sistema de control de versiones. Es un sistema distribuido en el que todos los nodos manejan la información en su totalidad y por lo tanto pueden actuar de cliente o de servidor en cualquier momento, eliminando así el concepto de centralizado. Guarda una copia entera de los datos con toda la estructura y los archivos necesarios (28).

- **SQLite**

Es un motor ligero de bases de datos de código abierto, caracterizado por mantener el almacenamiento de información persistente de forma sencilla. No requiere soporte de un servidor. SQLite no ejecuta un proceso

[8] Interfaz de programación de aplicaciones: Grupo de rutinas que provee un sistema operativo, representa una interfaz de comunicación entre componentes de software (43).

para administrar la información, implementa un conjunto de librerías encargadas de la gestión de la misma. No necesita configuraciones, libera al programador de todo tipo de configuraciones de puertos, tamaño, ubicación, entre otras tareas. Usa un archivo para el esquema completo de una base de datos, lo que permite aumentar la seguridad, ya que los datos de las aplicaciones *Android* no pueden ser accedidos por contextos externos. Es una herramienta de código abierto, disponible al dominio público de los desarrolladores al igual que los archivos de compilación e instrucciones de escalabilidad (29).

- ***Visual Paradigm 8.0***

Visual Paradigm es una herramienta CASE (Ingeniería de *Software* Asistida por Computación). La misma permite el modelado de diagramas para el desarrollo de programas informáticos. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de todo tipo de diagramas. Diseñado para la construcción de *software* de forma fiable a través de la utilización de un enfoque Orientado a Objetos (30).

1.6 Conclusiones parciales

En el capítulo se abordaron los elementos fundamentales relacionados con la gestión de tareas en Android. Además, partiendo del estudio de soluciones existentes de gestión de tareas en Android a nivel internacional fundamentalmente, se puede afirmar que los mismos no brindan una solución completa al problema existente, fundamentando la necesidad de desarrollar una aplicación que automatice las tareas, servicios y aplicaciones de los dispositivos móviles, optimizando a su vez sus recursos. Dicho estudio sirvió de apoyo a la investigación para lograr una mejor comprensión de estos sistemas y al análisis de la situación problemática antes planteada; aportó ideas para el diseño y desarrollo del sistema propuesto. La selección de las tecnologías, herramientas y metodología a utilizar permite establecer el entorno de desarrollo en el que se implementará la solución a la problemática antes planteada.

Capítulo 2: Análisis y diseño de la solución propuesta

2.1. Introducción

En el presente capítulo se describe la propuesta de solución. Se lleva a cabo la fase de Ejecución establecida por la metodología empleada. Además, se detallan artefactos asociados al análisis y diseño; especificando, entre otros elementos, requisitos funcionales, no funcionales y patrones de diseño.

2.2. Análisis y diseño

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Reflejan las necesidades de los clientes de un sistema que permita resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se denomina ingeniería de requerimientos.

2.3. Propuesta de solución

Partiendo de la problemática de la investigación se propone desarrollar una aplicación que permita gestionar a través de perfiles de configuración la conexión inalámbrica *wi-fi*, los perfiles de sonido y las aplicaciones de un dispositivo móvil con sistema operativo *Android*. Los perfiles estarán determinados por fechas y horas, con las cuales el dispositivo comparará con la hora actual del mismo para activar un perfil guardado, aplicándole así al equipo las configuraciones de los servicios que tiene almacenados en él mismo.

Se propone realizar una aplicación completa en vez de modificar una de las existentes puesto que las anteriores, realizan operaciones similares independientemente. También por la necesidad de corregir uno de los mayores defectos que poseen las aplicaciones estudiadas: los permisos de administración. Estas aplicaciones se encuentran disponibles en internet con un costo de descarga. En otro sentido, se desarrolla una nueva aplicación, porque para poder acceder a la información interna es necesario realizar una ingeniería inversa, lo que convertiría todo el proceso de desarrollo en un trabajo engorroso.

2.3.1 Requisitos funcionales (RF)

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de *software* que se desarrolle, de los posibles usuarios del *software* y del enfoque general tomado por la organización al redactar requerimientos (31).

RF1 Crear perfiles de configuración.
RF2 Seleccionar rango de hora del perfil de configuración.
RF3 Seleccionar las aplicaciones instaladas.
RF4 Activar/desactivar los servicios de conexión inalámbrica.
RF5 Mostrar las conexiones inalámbricas cercanas disponibles.
RF6 Seleccionar el perfil de audio.
RF7 Gestionar perfiles de configuración creados.
RF8 Mostrar las aplicaciones seleccionadas para el perfil de configuración deseado.

2.3.1.1 Historia de usuario

Una correcta descripción de los requisitos permitirá contar con un punto de vista más detallado en el momento de definir la arquitectura del sistema, estableciendo un punto de partida para las pruebas de funcionamiento y las futuras mejoras a incorporar. Las Historias de Usuario (HU) son el artefacto establecido por la metodología y el escenario que se emplea para la especificación o descripción de funcionalidades.

Tabla 2: Historia de Usuario: Crear perfil.

Número:HU1	**Nombre del HU:** Crear perfil
Programador: David Valdés Fernández	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: No posee riesgo	**Tiempo Real:** 1 semana
Descripción: Al accionar el botón para crear un perfil que se encuentra en el extremo inferior derecho de la pantalla (ver figura	

12), automáticamente la aplicación solicita el nombre deseado para el perfil a crear (ver figura 13), luego se muestran las opciones para adicionarle las fechas y horas en que se necesita que se active, el usuario deberá seleccionar el estado en que desea que la conexión inalámbrica se encuentre (activado/inactivado) y el audio del dispositivo móvil (vibrador-general-silencio). Una vez seleccionados todos los elementos que conforman un perfil, se presiona el botón para adicionar (ver figura 14) y automáticamente toda la información que falta por guardar se almacena en la base de datos.

-Fecha: En el espacio que muestra el título "Fecha" se encuentra un botón para adicionar fechas, la selección de las mismas se tiene que realizar de una a la vez, al realizar la elección, se presiona el botón de aceptar y automáticamente se guarda en la base de datos la fecha seleccionada.

-Hora: En el espacio que muestra el título "Hora" se encuentra un botón para adicionar las horas y minutos del día de la fecha seleccionada con anterioridad en que se desea que el perfil sea activado, inicialmente se selecciona la hora de inicio del perfil y luego la de fin, al terminar la elección automáticamente se almacena en la base de datos la información agregada.

- *Wi-fi*: En el extremo superior derecho de la pantalla del dispositivo se encuentra un ícono con el símbolo de la conexión inalámbrica w*i-fi*, al accionar sobre el ícono se cambiará el estado de la misma a través de valores de verdadero/falso, almacenándose en la base de datos una vez presionado el botón de aceptar el perfil a crear.

-Audio: En el extremo superior izquierdo se encuentra un ícono con el símbolo de audio, al accionar sobre el mismo, se mostrarán las opciones a seleccionar (vibrador-general-silencio), dependiendo de la selección se almacenará un valor entero una vez presionado el botón de aceptar el perfil a crear.

Prototipo de interfaz:

Figura 12: Prototipo de interfaz crear perfil.

Figura 13: Prototipo de interfaz nombre del perfil.

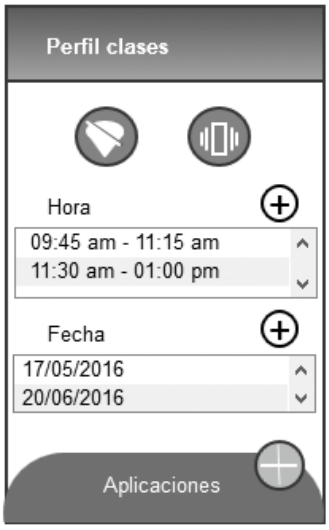


Figura 14: Prototipo de interfaz datos del perfil.

Tabla 3: Historia de usuario: Modificar perfil.

Número: HU2	**Nombre del requisito:** Modificar perfil

Programador: David Valdés Fernández	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: No posee riesgo	**Tiempo Real:** 1 semana

Descripción:

El sistema muestra una lista con los nombres almacenados de los perfiles, al accionar sobre el perfil deseado, se mostrará en pantalla toda la información que posee el perfil ya creado (ver figura 15). El usuario será capaz de realizar las modificaciones que estime conveniente sobre el perfil que se muestra en pantalla (ver figura 16).

-**Fecha**: Al accionar sobre la fecha se mostrará una lista con las fechas almacenadas para la activación del perfil, al hacer contacto con uno de los elementos que se encuentran en la lista, el mismo será eliminado de la lista y de la base de datos, a su vez se podrá adicionar una nueva fecha o se podrá mantener con los elementos restantes de la lista.

-Hora: Al accionar sobre la hora se mostrará una lista con las horas almacenadas para la activación del perfil, al hacer contacto con uno de los elementos que se encuentran en la lista, el mismo será eliminado de la lista y de la base de datos, a su vez se podrá adicionar una nueva fecha o se podrá mantener con los elementos restantes de la lista.

-*Wi-fi*: Al accionar sobre el ícono que se encuentra en el extremo superior izquierdo de la pantalla de la aplicación con el símbolo de wi-fi se podrá cambiar el valor almacenado para el mismo servicio.

-Audio: Al accionar sobre el ícono que se encuentra en el extremo superior derecho de la pantalla de la aplicación con el símbolo de audio se podrá cambiar el valor almacenado para el mismo servicio**.**

Prototipo de interfaz:

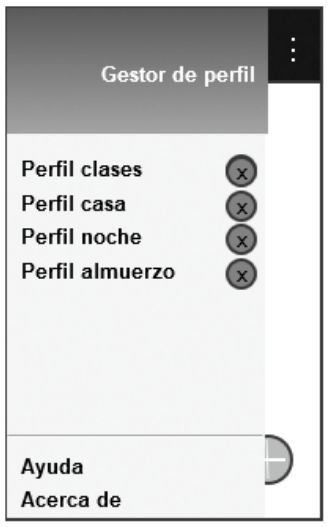


Figura 15: Prototipo de interfaz lista de perfiles.

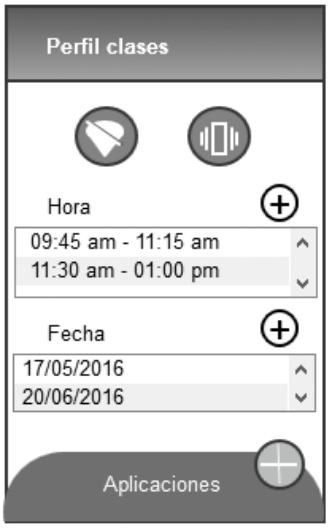


Figura 16: Prototipo de interfaz modificar perfil.

Tabla 4: Historia de usuario: Eliminar perfil.

Número: HU3	**Nombre del requisito:** Eliminar perfil

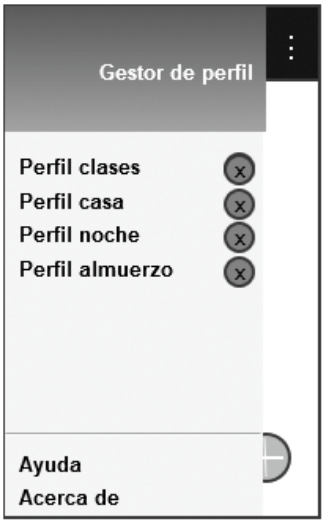
Programador: David Valdés Fernández	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1 día
Riesgo en Desarrollo: No posee riesgo	**Tiempo Real:** 1 semana
Descripción: El sistema muestra en pantalla todos los nombres de los perfiles que se encuentran almacenados en la base de datos de la aplicación, en el extremo derecho de los nombres se encuentra un ícono que al ser presionado elimina de la lista y de la base de datos el nombre del perfil y toda la información referente al mismo (ver figura 17).	
Prototipo de interfaz: 	

Figura 17: Prototipo de interfaz eliminar perfil.

Tabla 5: Historia de usuario: Activar perfil.

Número: HU4	**Nombre del requisito**: Activar perfil
Programador: David Valdés Fernández	**Iteración Asignada**: 1

Prioridad: Alta	**Tiempo Estimado**: 1 día
Riesgo en Desarrollo: No posee riesgo	**Tiempo Real**: 1 semana
Descripción: Al cumplirse todos los términos de fecha y hora de un perfil el sistema automáticamente activa el mismo, configurando los servicios del dispositivo móvil de acuerdo a la información guardada en el perfil.	
Prototipo de interfaz:	

2.3.2 Requerimientos no funcionales (RNF)

Los requerimientos no funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, define las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema (31).

Tabla 6: Requisitos no funcionales.

Recurso	Numeración	Descripción
Hardware	RNF1	El dispositivo debe poseer una capacidad de memoria RAM mínima de 512mb.
Software	RNF2	El dispositivo tiene que tener instalado sistema operativo Android.

	RNF3	La versión del sistema operativo tiene que ser 4.0 o mayor.
Interfaz	RNF4	La aplicación utilizará para la comodidad del usuario Material Design.

2.4. Arquitectura de *software*

Una arquitectura de *software* es una descripción de cómo se organiza un sistema informático. Las propiedades rendimiento, seguridad y disponibilidad, están influenciadas por la arquitectura utilizada. La misma afecta el desempeño y la potencia, así como la capacidad de distribución y mantenimiento de un sistema (32).

2.4.1 Arquitectura n-capas

La arquitectura en capas soporta el desarrollo incremental del sistema. Conforme se desarrolla una capa, alguno de los servicios proporcionados por esta deben quedar a disposición de los usuarios. La arquitectura también es cambiable y portátil. En tanto su interfaz no varía, una capa puede sustituirse por otra equivalente. Cuando una capa cambia o se agregan nuevas funcionalidades, solo resulta afectada la capa adyacente (32).

Organiza el sistema en capas con funcionalidades relacionadas con cada capa. Una capa ofrece servicios a la capa inmediatamente superior, de modo que las capas de nivel inferior representan servicio núcleo.

La solución propuesta será una versión inicial. Planteada esta situación, se decide emplear una arquitectura en capas. Teniendo en cuenta las futuras necesidades de los usuarios, se podrán modificar cualquiera de sus componentes sin tener que afectar en gran medida el resto de los mismos, solo los que se encuentran en su nivel.

2.5. Patrón arquitectónico

Descripción abstracta de una arquitectura de *software* que se ensayó y puso a prueba en algunos sistemas de *software* distintos. La descripción del patrón incluye información acerca de dónde es adecuado usar el patrón y la organización de los componentes de la arquitectura (32).

Modelo-Vista-Controlador

Separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí. El componente Modelo maneja los datos del sistema y las operaciones asociadas a esos datos. El componente Vista define y gestiona como se presentan los datos al usuario. El componente Controlador dirige la interacción del usuario y pasa estas interacciones a Vista y Modelo (32).

Este patrón permite que los datos cambien de manera independiente de su representación y viceversa. Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos.

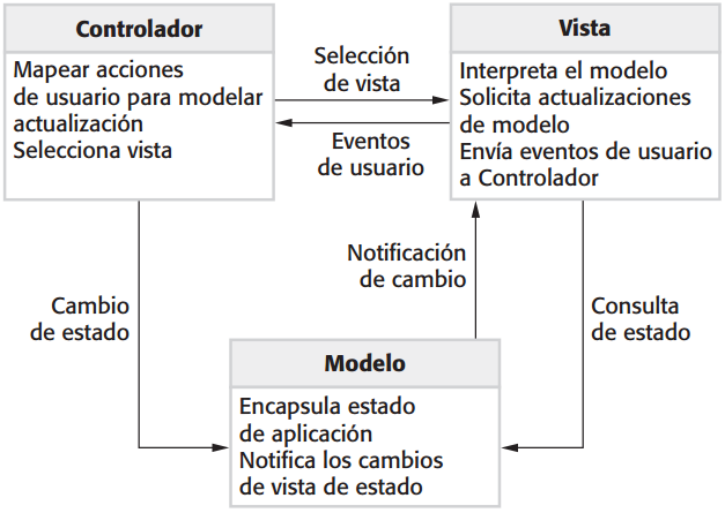


Figura 18: Patrón arquitectónico Modelo-Vista-Controlador.

La solución propuesta constará de tres capas. Dentro de la capa Vista se encontrarán los archivos visuales con los que el usuario interactuará directamente, introduciendo las configuraciones pertinentes y resolviendo sus necesidades. En la capa Modelo se tendrán las clases que interactúan con las tablas de base de datos, lugar donde se almacenarán todos los perfiles, cambios realizados y otras informaciones que son necesarias almacenar. La capa Controlador contendrá las clases que interactúan con la capa Vista, recibiendo las solicitudes de eventos de los usuarios, y con la capa Modelo, registrando los cambios realizados por el mismo.

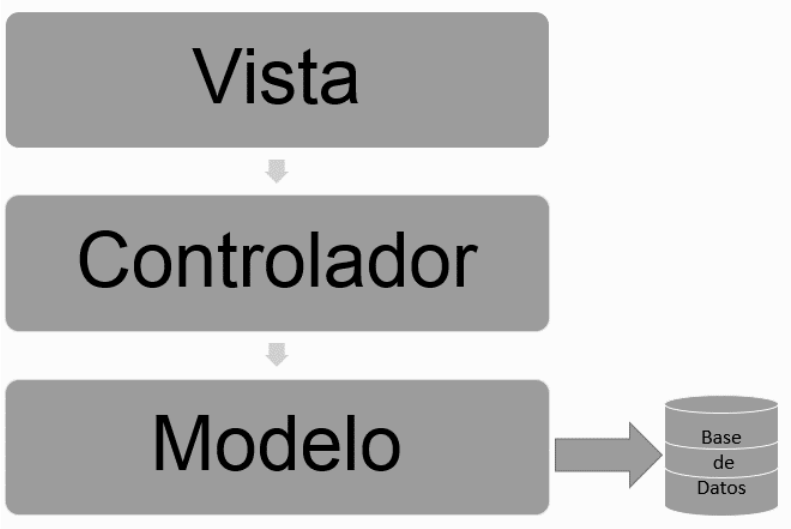


Figura 19: Patrón arquitectónico Modelo-Vista-Controlador de la solución propuesta.

2.6. Patrones de diseño

Los patrones de diseño son modelos que brindan soluciones al problema de cómo enfrentar un nuevo programa o proyecto. Los patrones capturan las experiencias y las hacen accesibles a los no expertos, ayudando también a la comunicación entre programadores.

Los patrones de diseño GRASP[9] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (33).

Patrones GRASP:

Experto: es el patrón más empleado para asignar responsabilidades. Consiste en asignar una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo (33). En la aplicación se evidencia el uso del patrón experto en las clases PerfilActivity y GServActivity. En estas clases se puede observar el empleo de dicho patrón cuando se va a crear un perfil, activar un perfil, modificar un perfil y eliminar un perfil.

[9] GRASP: *General Responsability Software Patterns*

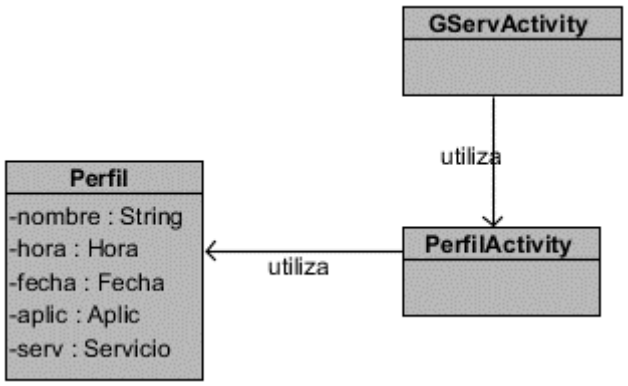


Figura 20: Patrón experto.

Alta cohesión: este patrón muestra lo relacionadas que se encuentran las responsabilidades de una clase, o una clase con responsabilidades altamente relacionadas y que no lleva a cabo gran cantidad de trabajo (33). Este patrón se evidencia en la clase GServActivity, cuyas funcionalidades colaboran para gestionar un perfil, para lo cual utiliza la clase PerfilActivity.

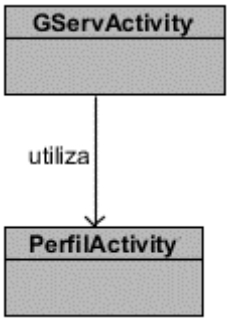


Figura 21: Patrón alta cohesión.

Bajo Acoplamiento: el acoplamiento indica que tan fuerte se encuentra una clase conectada con otra o influye sobre la misma, una clase con bajo acoplamiento no depende de otras clases (33). Ejemplo del uso de este patrón en la solución propuesta se muestra en la clase PerfilActivity, Fecha y Hora, donde se minimizan las relaciones de estas con el resto de las clases.

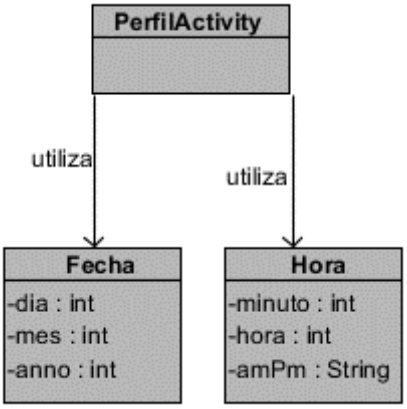


Figura 22: Patrón bajo acoplamiento.

Creador: el patrón creador indica cuales son las clases que tienen la responsabilidad de crear una nueva instancia de una clase. Ejemplo del uso del patrón creador es la clase PerfilActivity, la misma es la responsable de agregar, contiene, almacena y emplea objetos de la clase Perfil, tiene los datos necesarios para inicializar a la clase Perfil cuando esta es creada (33).

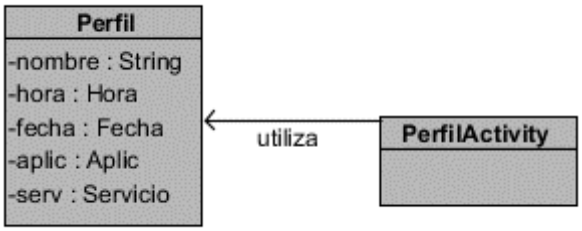


Figura 23: Patrón creador.

2.7. Diagrama de clases

Los diagramas de clases representan las relaciones existentes entre las clases de un sistema. Se emplea para modelar la vista estática del diseño de un sistema, visualizar, especificar y documentar modelos estructurales, y para construir sistemas ejecutables a través de ingeniería directa o inversa (34).

La estructura que guiará el desarrollo de la aplicación para la gestión de perfiles en sistemas operativo *Android* tendrá como base el siguiente diagrama de clases:

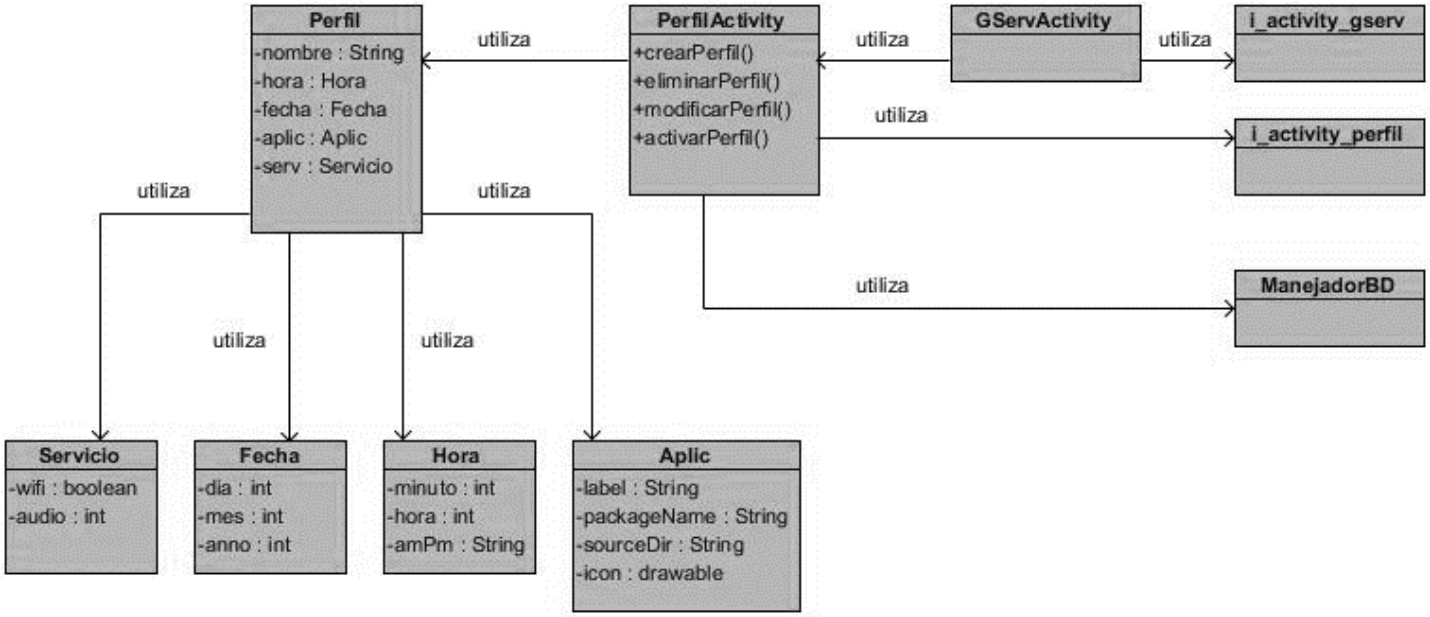


Figura 24: Diagrama de clases.

2.8. Conclusiones parciales

Con el uso de la arquitectura en capas y el empleo de patrones GRASP se garantizó un mejor entendimiento de la solución propuesta, se sentaron las bases para su implementación, favoreciendo a los programadores con conocimientos referentes a sistemas operativos *Android*. Las historias de usuarios agrupan todos los requerimientos en cuatro funciones, presentadas en un lenguaje natural, brindando una mayor comprensión de los mismos.

Capítulo 3: Implementación y pruebas

3.1 Introducción

En el presente capítulo se exponen aspectos asociados a la implementación de la aplicación. Se registran los resultados obtenidos de la aplicación de pruebas al sistema, con el fin de validar el cumplimiento de los requerimientos establecidos. Se presentan las pautas de codificación y los resultados obtenidos de las pruebas en busca de errores en la aplicación.

3.2 Implementación

Una vez definidas las HU y concluido el diseño de la aplicación tiene lugar la codificación de la solución propuesta, cuyos objetivos están encaminados a desarrollar de manera iterativa e incremental un producto completo y listo para el despliegue.

3.2.1 Pautas de codificación

Los estándares de codificación garantizan la comunicación entre los programadores, permitiendo la reutilización y el mantenimiento de los sistemas. Durante el proceso de desarrollo de la propuesta de solución se emplea el estándar *CamelCase*[10] para la definición de variables y métodos con el fin de lograr un buen entendimiento del código, el mismo se divide en dos estándares *lowerCamelCase*[11] y *UpperCamelCase*[12].

- **Comentarios en las funciones**

Las funciones deben tener un comentario antes de ser declaradas, explicando el funcionamiento de las mismas.

[10] *CamelCase*: Estándar de codificación que define el comienzo con una letra minúscula y las siguientes iniciales de palabras con mayúsculas
[11] *lowerCamelCase*: Estándar de codificación *CamelCase* que comienza con letra minúscula y el resto de las palabras las comienzan con mayúscula.
[12] *UpperCamelCase:* Estándar de codificación *CamelCase* en el que todas las palabras comienzan con letra mayúscula.

```
/*Mostrar las fechas y horas que estan a punto de ser guardadas para un perfil determinado*/
    public android.app.Dialog mostrarFechas(int i) {
```

- **Nombres de las variables en Java**

Los nombres de variables emplean el estándar de codificación *lowerCamelCase*, deben ser cortos y descriptivos. Al leer el nombre el programador debe ser capaz de identificar para que se emplea. Pueden tener solo caracteres alfanuméricos.

```
private ArrayList<Fecha> listFecha = new ArrayList<Fecha>();
```

- **Nombres de las variables en XML**

Los nombres de las variables en los archivos XML se definen en minúsculas todas las letras, comienzan con las iniciales del nombre del objeto seguido por *underscore* (“_”) y luego por un nombre corto y descriptivo.

```
  android:id="@+id/add_fecha"
```

- **Nombres de las variables referentes a la base de datos**

Los nombres de las variables se definen en minúsculas y al final el nombre del *framework* (ADA) empleado en mayúscula.

```
 private ADAfecha dateADA;
```

- **Nombres de los métodos**

Los nombres de los métodos se definen con el estándar de codificación *lowerCamelCase*. Pueden tener solo caracteres alfanuméricos.

```
protected void onCreate(Bundle savedInstanceState) {
```

- **Nombres de las clases**

En los nombres de las clases se emplea el estándar de codificación *UpperCamelCase*.

```
public class PerfilActivity extends AppCompatActivity implements OnClickListener {
```

- **Base de datos**

En la base de datos las clases comienzan con el nombre del *framework* empleado en mayúscula ADA, seguido de minúsculas.

```
public class ADAperfil extends Entity {
```

Los nombres de las tablas y los métodos se escriben en minúsculas.

```
@TableField(name = "nombre", datatype = Entity.DATATYPE_STRING)
private String nombre;
```

3.3 Pruebas

El desarrollo de sistemas de *software* implica la realización de actividades predispuestas a encontrar errores. Las pruebas requieren que se descarten ideas sobre la calidad o corrección del sistema. Los resultados obtenidos se registran para realizar un proceso de evaluación en el que los mismos se comparan con los resultados esperados de la aplicación, con el fin de poder distribuir un *software* con toda la calidad que requiere.

La metodología empleada establece la realización de tres tipos de pruebas. De ellas se aplicarán las pruebas internas y de aceptación a la solución implementada.

3.3.1 Pruebas de aceptación

Las pruebas de aceptación comprueban los errores a través del análisis de los requisitos, proponen realizar numerosas operaciones sobre la interfaz visual del *software* con el fin de saber si se cumplen todos los requisitos (35).

Los casos de pruebas de aceptación pretenden demostrar que:

- Las funciones del *software* son operativas.

- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Se pretende encontrar con este tipo de prueba los siguientes errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en acceso a base de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Para comprobar el buen funcionamiento o la existencia de errores del software, se publicó la aplicación en el sitio web humanos.uci.cu en la dirección http://humanos.uci.cu/2016/06/compartetusoftware-aplicacion-android-para-la-gestion-de-tareas/#comment-79787. En un plazo de 11 horas, la aplicación tuvo un total de 295 visitas y 17 comentarios, planteando los criterios de los usuarios referentes al software. Entre las publicaciones no se encontró ninguna insatisfacción o duda en el funcionamiento de la aplicación. Las principales ideas se centraron en la necesidad de una versión con la posibilidad de activar los perfiles de configuración de acuerdo a la ubicación geográfica, específicamente dentro de la universidad, permitiendo la gestión de servicios y aplicaciones para las clases, reuniones y otros eventos. Los usuarios exponen la libertad que proporciona GesPer al permitir realizar la vida cotidiana sin preocupaciones sobre el dispositivo móvil (ver anexo 2). A partir del análisis anterior se comprobó la aceptación causada por la aplicación a los visitantes del sitio web, corroborando así el buen funcionamiento del *software*.

3.3.2 Pruebas unitarias

Las pruebas unitarias basan su funcionamiento en el análisis del código interno del *software*. Desarrollar estas pruebas permite garantizar que las operaciones se ajustan a las especificaciones, y que todos los componentes se han probado de forma adecuada.

En las pruebas unitarias se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, comprobando los bucles y condiciones, y comprobando el estado del programa

en varios puntos. Es un método de diseño de casos de pruebas que usa la estructura de control del diseño procedimental para derivar los casos de pruebas (35).

Garantizan que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecutan todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

Para el análisis de la aplicación a través de pruebas unitarias, se empleó el método manual Camino Básico. El mismo consiste en reconocer los diferentes caminos que puede tomar un método para obtener una solución, analiza cada uno de los caminos, comprobando que no existan errores y sea recorrido de manera completa y eficiente. Analizando los dos métodos fundamentales “ActivarPerfil” y “CrearPerfil”, se detectó seis caminos básicos entre los dos. Aplicando la métrica Complejidad Ciclomática se detectó para el primer método un total de seis pruebas para analizarlo en su totalidad y para el segundo tres pruebas, teniendo un total de nueve pruebas para un análisis completo de los dos métodos. En busca de una solución óptima, se realizaron tres iteraciones de pruebas.

Tabla 7: Resultados de pruebas unitarias.

Método	**Caminos**	**Pruebas**	**Iteración 1**		**Iteración 2**		**Iteración 3**	
			S	NS	S	NS	S	NS
ActivarPerfil	4	6	4	2	5	1	6	0
CrearPerfil	2	3	1	2	3	0	3	0
Total	**6**	**9**	**5**	**4**	**8**	**1**	**9**	**0**

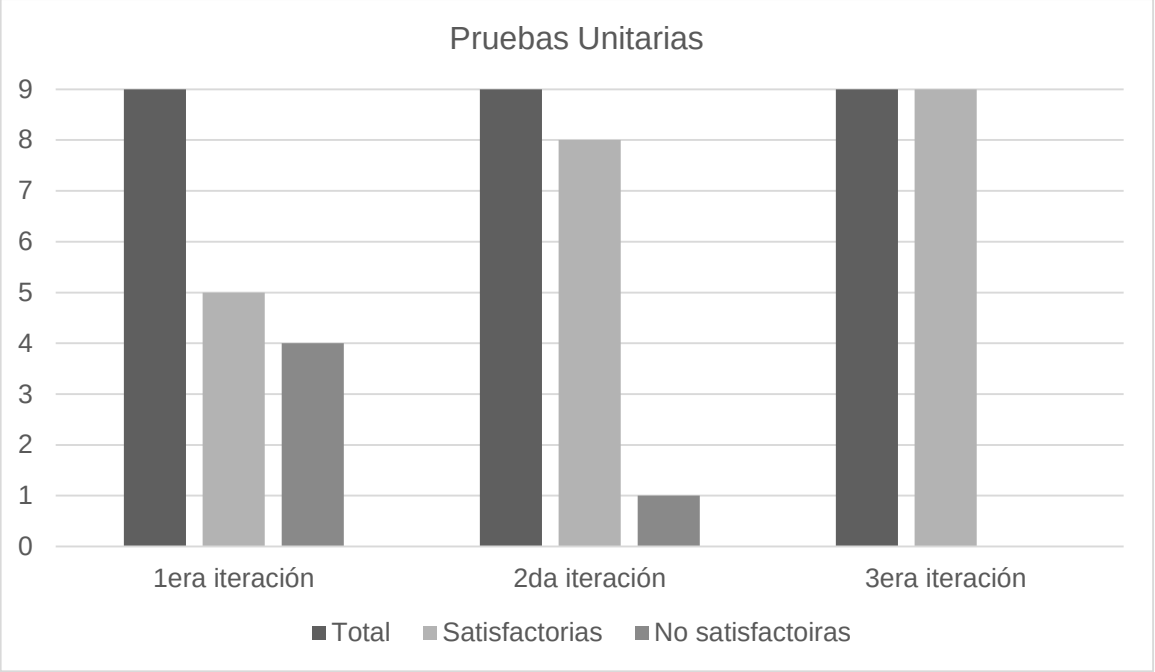


Figura 25: Resultados de las pruebas unitarias.

3.3.3. Pruebas de rendimiento

Con la necesidad de comprobar el principal problema que se intenta solucionar con el desarrollo de la aplicación, el consumo excesivo de recursos del dispositivo móvil en el cual pueda estar instalada, se realizaron pruebas de rendimiento, ya que las pruebas determinadas por la metodología de desarrollo de software AUP-UCI no valida el consumo de recursos planteados en el primer capítulo. Para ello se ejecutaron tres procesos de experimentación. El primero encaminado a comprobar el consumo de batería del dispositivo móvil, con un conjunto de las aplicaciones que se estudiaron con anterioridad. El segundo está dirigido a verificar el consumo de RAM, empleando el mismo conjunto de aplicaciones. El tercero se enfoca en analizar el funcionamiento de la CPU.

Experimentación # 1

Las aplicaciones instaladas en el dispositivo para abarcar un gran número de servicios y acciones posibles, fueron *Automate, MacroDroid* y *Trigger.* Se realizó un estudio por un período de tres días analizando el porciento de consumo por aplicación y el consumo general del dispositivo. En temas de porcentaje:

Tabla 8: Consumo de batería.

Aplicación	***Consumo (%)***
Automate- MacroDroid- Trigger	*84 – 46*
GesPer	*92 – 65*

En un período de uso de las aplicaciones de tres días el estado de la batería descendió de 84% a 46%. Una vez desinstaladas las mismas e instalando la aplicación desarrollada, en un período de tiempo de tres días y en el mismo dispositivo, la batería descendió 92% a 65. Concluyendo es estudio anterior, el consumo de batería disminuyó empleando el sistema GesPer.

Experimentación # 2

Empleando las opciones que brinda el sistema se analizó el consumo de RAM de cada una de las aplicaciones en el dispositivo. Al accionar sobre los ajustes de sistema, se muestran un conjunto de elementos a configurar, entre ellos se encuentran las aplicaciones. En uno de los espacios para la configuración y gestión de las aplicaciones se puede observar el consumo de memoria. A continuación, se muestra una tabla en la que se resumió el consumo de estas aplicaciones en megabytes (MB):

Tabla 9: Consumo de memoria RAM.

Aplicación	**Consumo (MB)**
Automate	8
MacroDroid	5
Trigger	6
GesPer	5

Experimentación # 3

Con el dispositivo móvil inactivo solo permitiendo correr en segundo plano los servicios que vienen por defecto y los de las tres aplicaciones instaladas, el estado de la CPU oscila entre los valores de 6% y 10%. En el caso

de tener instalado en el equipo solamente la aplicación GesPer, con sus servicios ejecutándose en segundo plano, el estado de la CPU variaba entre los valores 4% y 7%.

Análisis de los resultados de la experimentación

Teniendo en cuenta los resultados obtenidos durante los procesos de experimentación, se arriba a las siguientes conclusiones:

- En términos de consumo de energía, el tener varias aplicaciones ejecutando servicios en segundo plano tiene la desventaja de mantener un alto consumo de energía. Estas aplicaciones realizan tareas distintas, pero poseen funciones semejantes, ejecutando acciones sobre los mismos servicios, produce un mayor consumo con respecto a una aplicación que agrupa en su uso estos servicios y funcionalidades.
- Al tener un conjunto de aplicaciones instaladas que poseen tareas ejecutadas en segundo plano, se mantiene un consumo elevado de memoria RAM, dificultando el funcionamiento de otras aplicaciones y del mismo dispositivo.
- De igual manera, el consumo de la CPU se mantiene por encima al emplear las tres aplicaciones debido a que todas poseen acciones ejecutadas permanentemente.

3.4 Conclusiones parciales

En este capítulo se describió con detalles el proceso de desarrollo de la aplicación para la gestión de perfiles de configuración en dispositivos móviles con sistemas operativos *Android*, dando como resultado un *software* listo para su distribución. El establecimiento de las pautas de codificación permitió el desarrollo de un código reutilizable, legible y específico para el programador. El proceso llevado a cabo y las pruebas realizadas al sistema permitieron validar y comprobar que la aplicación cumple con los requisitos funcionales presentados para la implementación de la misma.

Conclusiones generales

Entre los principales resultados obtenidos durante la investigación y desarrollo del producto final se puede destacar que:

- El estudio realizado a herramientas con funcionalidades similares facilitó la definición de una propuesta de solución que realiza configuraciones sobre los servicios y aplicaciones en dispositivos con SO *Android* utilizando menor cantidad de recursos que las estudiadas.
- El empleo de las herramientas, metodología y tecnologías seleccionadas permitieron analizar, describir e implementar la solución propuesta, cumpliendo con los requisitos planteados.
- La implementación de la aplicación para la gestión de perfiles de configuración, luego de ser validada a partir de las pruebas de rendimiento y las definidas por la metodología AUP-UCI, permitió el correcto funcionamiento y configuración de servicios en los dispositivos móviles con SO *Android*.

Recomendaciones

Aunque la aplicación cumple con los requisitos planteados, y brinda un mayor confort a los posibles usuarios de la misma, los resultados pueden mejorarse y garantizar una mayor configuración del dispositivo móvil. Por ello se recomienda:

- Perfeccionar el método de activación de perfiles agregando servicios de ubicación geográfica.
- Incorporar opciones como envío de mensajes y correos programados para un perfil de configuración.
- Permitir visualizar los mensajes y correos recibidos desde la ventana flotante.
- Permitir el intercambio de perfiles ya creados con otros usuarios.
- Desarrollar la aplicación “GesPer” para ser utilizada en otros sistemas operativos.

Bibliografía

1. PuroMarketing. *Nuevos datos demuestran la dependencia a los dispositivos móviles.* [En línea] 2015. [Citado el: 8 de diciembre de 2015.] http://www.puromarketing.com/12/19353/datos-demuestran-dependencia-dispositivos-moviles.html.

2. International Data Corporation. IDC Analyze the Future. *Smartphone OS.* [En línea] 2015. [Citado el: 16 de noviembre de 2015.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

3. Orosco, David. Definición de Android. [En línea] 2014. [Citado el: 30 de noviembre de 2015.] http://conceptodefinicion.de/android/.

4. El Centro de Tesis, Documentos, Publicaciones y Recursos Educativos más amplio de la Red. sbweb. *webcenter.* [En línea] [Citado el: 24 de mayo de 2016.] http://www.sc.ehu.es/sbweb/webcentro/automatica/WebCQMH1/PAGINA%20PRINCIPAL/Automatizacion/Automatizacion.htm.

5. Barrios, Alex. FuturisticTecno. *Root. Definición, ventajas y desventajas. Al final, es tu decisión. .* [En línea] 1 de marzo de 2012. [Citado el: 16 de diciembre de 2015.] http://futuristictecno.blogspot.com/2012/03/root-definicion-ventajas-y-desventajas.html.

6. Universidad de Jaén. Servicio de informática. *¿Qué es un perfil?* [En línea] [Citado el: 15 de diciembre de 2015.] http://faq.ujaen.es/index.php?action=artikel&cat=24&id=35&artlang=es.

7. Microsoft. Microsoft. *¿Qué son los perfiles de usuario?* [En línea] 2015. [Citado el: 15 de diciembre de 2015.] http://windows.microsoft.com/es-419/windows-vista/what-are-user-profiles.

8. Sicilia, Miguel Angel. Openstax cnx. *Disparadores en bases de datos relacionales.* [En línea] 2015. [Citado el: 16 de diciembre de 2015.] http://cnx.org/contents/OfSwDu71@1/Disparadores-en-bases-de-datos.

9. Mejía, Luis Miguel Arteaga. El sistema operativo GNU. *¿Qué es el software libre?* [En línea] 5 de septiembre de 2015. [Citado el: 16 de diciembre de 2015.] http://www.gnu.org/philosophy/free-sw.es.html.

10. Alegsa.com.ar. *Diccionario de informática y tecnología .* [En línea] 16 de mayo de 2010. [Citado el: 8 de diciembre de 2015.] http://www.alegsa.com.ar/Dic/codigo%20fuente.php.

11. EcuRed Conocimiento con todos y para tdos. [En línea] 8 de diciembre de 2015. [Citado el: 19 de noviembre de 2015.] http://www.ecured.cu/C%C3%B3digo_abierto.

12. Falsal, Adnan. Descargar Play Store. *¿Qué es una APK?* [En línea] 26 de abril de 2015. [Citado el: 30 de noviembre de 2015.] http://descargarplaystore.mobi/que-es-una-apk/.

13. Uribe, Manuel Echeverry. Un Geek en Colombia. *Móviles.* [En línea] 4 de febrero de 2015. [Citado el: 26 de noviembre de 2015.] http://www.ungeekencolombia.com/5-excelentes-aplicaciones-para-automatizar-android/.

14. Martino, Gonzalo. GetMóvil. *MacroDroid: Automatiza las Tareas Importantes de tu Android (APK).* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] https://getmovil.com/playstore/macrodroid-android/.

15. Descargadictos. *Tasker 4.4 Final Pro [Automatización Total] [Apk] [Android] [MG-ZS].* [En línea] [Citado el: 9 de diciembre de 2015.] http://descargadictos.info/downloads/222815/tasker-4-4-final-pro-automatizacion-total-apk-android-mg-zs.html.

16. Ramirez, Manuel. Android SIS. *Crear tareas automatizadas con diagramas en Android con Automate.* [En línea] 27 de septiembre de 2014. [Citado el: 9 de diciembre de 2015.] http://www.androidsis.com/crea-tareas-automatizadas-con-diagramas-en-android-con-automate/.

17. Tello, Edwin. HostDime Blog. *10 Aplicaciones Gratis Para Automatizar Tareas En Android.* [En línea] 2015. [Citado el: 26 de noviembre de 2015.] http://blog.hostdime.com.co/10-aplicaciones-gratis-para-automatizar-tareas-en-android/.

18. NeoTeo. *7 aplicaciones gratuitas para automatizar tareas (Android).* [En línea] 2015. [Citado el: 26 de noviembre de 2015.] http://www.neoteo.com/7-aplicaciones-gratuitas-para-automatizar-tareas-a.

19. Martín, Raúl. Mi bq y yo. *Wi-Fi Matic gestiona el Wi-Fi por ti.* [En línea] 29 de mayo de 2014. [Citado el: 9 de diciembre de 2015.] http://www.mibqyyo.com/articulos/2014/05/29/wi-fi-matic-gestiona-el-wi-fi-por-ti/#/vanilla/discussion/embed/?vanilla_discussion_id=0.

20. Android Experto. *Las mejores aplicaciones para gestionar aplicaciones en Android.* [En línea] 7 de julio de 2014. [Citado el: 9 de diciembre de 2015.] http://www.androidexperto.com/aplicaciones-android/las-mejores-aplicaciones-gestionar-aplicaciones-android/.

21. Valdéz, José Luis Cendejas. ecumed.net Enciclopedia Virtual. *Modelos y metodologías para el desarrollo de software.* [En línea] 2014. [Citado el: 19 de noviembre de 2015.] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

22. *Gestión de la Calidad en el Ciclo de Desarrollo del Software de proyectos.* Acebo, Yadira Bagarotti. 2, Granma : s.n., 2012, Vol. 16. ISSN 1027-975X.

23. Sánchez, Tamara Rodríguez. *Metodología de Desarrollo para la Actividad Productiva.* La Habana : s.n.

24. Solís, Carlos. *Android Studio.* s.l. : Video2Brain.

25. Hernandez, Javier. Android Hispano. [En línea] junio de 2013. [Citado el: 3 de diciembre de 2015.] http://android-hispano.blogspot.com/2013/06/guia-de-usuario-del-complemento-gradle.html.

26. González, Antony García. Panama Hitek. [En línea] 15 de junio de 2015. [Citado el: 8 de diciembre de 2015.] http://panamahitek.com/que-es-maven-y-para-que-se-utiliza/.

27. Salvioni, Nathalie M. Aquino y Acosta, Juan Carlos Frutos. *Fundamentos de la Máquina Virtual Java y el Entorno .Net .* 2002.

28. Andrearrs. Por qué Git es el sistema de control de versiones más popular. [En línea] 12 de mayo de 2014. [Citado el: 8 de diciembre de 2015.] file:///C:/Users/ds/AppData/Roaming/Mozilla/Firefox/Profiles/402230nc.dev-edition-default/ScrapBook/data/20151204164135/index.html.

29. Revelo, James. Hermosa Programación. *Tutorial De Bases De Datos SQLite En Aplicaciones Android.* [En línea] 14 de octubre de 2014. [Citado el: 8 de diciembre de 2015.] http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/.

30. EcuRed. [En línea] 2002. [Citado el: 8 de dicimbre de 2015.] http://www.ecured.cu/Visual_Paradigm. ISBN: 8448132149..

31. Somerville. Materiales_basicos. *eva.uci.cu.* [En línea] [Citado el: 10 de 2 de 2016.] http://eva.uci.cu/file.php/161/Documentos/Materiales_basicos/Materiales_basicos_de_la_Unidad_4/Sommerville/Sommerville_Parte_II_Requerimientos.pdf).

32. —. *Ingenieria de software.* México : s.n., 2011.

33. Ingenieria de Software. *Patrones de diseño.* [En línea] 2011. [Citado el: 14 de marzo de 2016.] http://www.cartagena99.com/recursos/alumnos/apuntes/Patrones%20de%20Diseno.pdf.

34. Aria, Miriela Velazquez. *Componente de medición de la calidad de imágenes de huellas dactilares.* La Habana : s.n., 2014.

35. *Técnicas de pruebas.* 2015.

36. ProcesosdeSoftware. *Metodología XP.* [En línea] 2015. [Citado el: 8 de diciembre de 2015.] https://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

37. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* Romero, Gladys Marsi Peñalver, Puente, Sergio Jesus García De La y Abad, Abel Meneses. La Habana : s.n., 2010.

38. Ordenadores y Portátiles. *Funciones PDA y como nos pueden ayudar estos dispositivos.* [En línea] 2014. [Citado el: 9 de diciembre de 2015.] http://www.ordenadores-y-portatiles.com/funciones-pda.html.

39. Definición.de. *Definición de perfil.* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] http://definicion.de/perfil/.

40. DefiniciónABC. *Memoria RAM.* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] http://www.definicionabc.com/tecnologia/memoria-ram.php.

41. DefiniciónABC. *Definición de CPU.* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] http://www.definicionabc.com/tecnologia/cpu.php.

42. carlospes.com. *Diccionario de Informática.* [En línea] 2014. [Citado el: 16 de diciembre de 2015.] http://www.carlospes.com/minidiccionario/pseudocodigo.php.

43. Algesa.com.ar. *Diccionario de informática y tecnología.* [En línea] 30 de mayo de 2013. [Citado el: 16 de diciembre de 2015.] http://www.alegsa.com.ar/Dic/api.php.

44. Alegsa.com.ar. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA.* [En línea] 2015. [Citado el: 16 de diciembre de 2015.] http://www.alegsa.com.ar/Dic/script.php.

45. Alegsa, Leandro. ALEGSA.com.ar. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA.* [En línea] 12 de mayo de 2010. [Citado el: 15 de dicuembre de 2015.] http://www.alegsa.com.ar/Dic/gps.php.

Referencias

1. PuroMarketing. *Nuevos datos demuestran la dependencia a los dispositivos móviles.* [En línea] 2015. [Citado el: 8 de diciembre de 2015.] http://www.puromarketing.com/12/19353/datos-demuestran-dependencia-dispositivos-moviles.html.

2. International Data Corporation. IDC Analyze the Future. *Smartphone OS.* [En línea] 2015. [Citado el: 16 de noviembre de 2015.] http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

3. Orosco, David. Definición de Android. [En línea] 2014. [Citado el: 30 de noviembre de 2015.] http://conceptodefinicion.de/android/.

4. El Centro de Tesis, Documentos, Publicaciones y Recursos Educativos más amplio de la Red. sbweb. *webcenter.* [En línea] [Citado el: 24 de mayo de 2016.] http://www.sc.ehu.es/sbweb/webcentro/automatica/WebCQMH1/PAGINA%20PRINCIPAL/Automatizacion/Automatizacion.htm.

5. Barrios, Alex. FuturisticTecno. *Root. Definición, ventajas y desventajas. Al final, es tu decisión. .* [En línea] 1 de marzo de 2012. [Citado el: 16 de diciembre de 2015.] http://futuristictecno.blogspot.com/2012/03/root-definicion-ventajas-y-desventajas.html.

6. Universidad de Jaén. Servicio de informática. *¿Qué es un perfil?* [En línea] [Citado el: 15 de diciembre de 2015.] http://faq.ujaen.es/index.php?action=artikel&cat=24&id=35&artlang=es.

7. Microsoft. Microsoft. *¿Qué son los perfiles de usuario?* [En línea] 2015. [Citado el: 15 de diciembre de 2015.] http://windows.microsoft.com/es-419/windows-vista/what-are-user-profiles.

8. Sicilia, Miguel Angel. Openstax cnx. *Disparadores en bases de datos relacionales.* [En línea] 2015. [Citado el: 16 de diciembre de 2015.] http://cnx.org/contents/OfSwDu71@1/Disparadores-en-bases-de-datos.

9. Mejía, Luis Miguel Arteaga. El sistema operativo GNU. *¿Qué es el software libre?* [En línea] 5 de septiembre de 2015. [Citado el: 16 de diciembre de 2015.] http://www.gnu.org/philosophy/free-sw.es.html.

10. Alegsa.com.ar. *Diccionario de informática y tecnología .* [En línea] 16 de mayo de 2010. [Citado el: 8 de diciembre de 2015.] http://www.alegsa.com.ar/Dic/codigo%20fuente.php.

11. EcuRed Conocimiento con todos y para tdos. [En línea] 8 de diciembre de 2015. [Citado el: 19 de noviembre de 2015.] http://www.ecured.cu/C%C3%B3digo_abierto.

12. Falsal, Adnan. Descargar Play Store. *¿Qué es una APK?* [En línea] 26 de abril de 2015. [Citado el: 30 de noviembre de 2015.] http://descargarplaystore.mobi/que-es-una-apk/.

13. Uribe, Manuel Echeverry. Un Geek en Colombia. *Móviles.* [En línea] 4 de febrero de 2015. [Citado el: 26 de noviembre de 2015.] http://www.ungeekencolombia.com/5-excelentes-aplicaciones-para-automatizar-android/.

14. Martino, Gonzalo. GetMóvil. *MacroDroid: Automatiza las Tareas Importantes de tu Android (APK).* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] https://getmovil.com/playstore/macrodroid-android/.

15. Descargadictos. *Tasker 4.4 Final Pro [Automatización Total] [Apk] [Android] [MG-ZS].* [En línea] [Citado el: 9 de diciembre de 2015.] http://descargadictos.info/downloads/222815/tasker-4-4-final-pro-automatizacion-total-apk-android-mg-zs.html.

16. Ramirez, Manuel. Android SIS. *Crear tareas automatizadas con diagramas en Android con Automate.* [En línea] 27 de septiembre de 2014. [Citado el: 9 de diciembre de 2015.] http://www.androidsis.com/crea-tareas-automatizadas-con-diagramas-en-android-con-automate/.

17. Tello, Edwin. HostDime Blog. *10 Aplicaciones Gratis Para Automatizar Tareas En Android.* [En línea] 2015. [Citado el: 26 de noviembre de 2015.] http://blog.hostdime.com.co/10-aplicaciones-gratis-para-automatizar-tareas-en-android/.

18. NeoTeo. *7 aplicaciones gratuitas para automatizar tareas (Android).* [En línea] 2015. [Citado el: 26 de noviembre de 2015.] http://www.neoteo.com/7-aplicaciones-gratuitas-para-automatizar-tareas-a.

19. Martín, Raúl. Mi bq y yo. *Wi-Fi Matic gestiona el Wi-Fi por ti.* [En línea] 29 de mayo de 2014. [Citado el: 9 de diciembre de 2015.] http://www.mibqyyo.com/articulos/2014/05/29/wi-fi-matic-gestiona-el-wi-fi-por-ti/#/vanilla/discussion/embed/?vanilla_discussion_id=0.

20. Android Experto. *Las mejores aplicaciones para gestionar aplicaciones en Android.* [En línea] 7 de julio de 2014. [Citado el: 9 de diciembre de 2015.] http://www.androidexperto.com/aplicaciones-android/las-mejores-aplicaciones-gestionar-aplicaciones-android/.

21. Valdéz, José Luis Cendejas. ecumed.net Enciclopedia Virtual. *Modelos y metodologías para el desarrollo de software.* [En línea] 2014. [Citado el: 19 de noviembre de 2015.] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

22. *Gestión de la Calidad en el Ciclo de Desarrollo del Software de proyectos.* Acebo, Yadira Bagarotti. 2, Granma : s.n., 2012, Vol. 16. ISSN 1027-975X.

23. Sánchez, Tamara Rodríguez. *Metodología de Desarrollo para la Actividad Productiva.* La Habana : s.n.

24. Solís, Carlos. *Android Studio.* s.l. : Video2Brain.

25. Hernandez, Javier. Android Hispano. [En línea] junio de 2013. [Citado el: 3 de diciembre de 2015.] http://android-hispano.blogspot.com/2013/06/guia-de-usuario-del-complemento-gradle.html.

26. González, Antony García. Panama Hitek. [En línea] 15 de junio de 2015. [Citado el: 8 de diciembre de 2015.] http://panamahitek.com/que-es-maven-y-para-que-se-utiliza/.

27. Salvioni, Nathalie M. Aquino y Acosta, Juan Carlos Frutos. *Fundamentos de la Máquina Virtual Java y el Entorno .Net .* 2002.

28. Andrearrs. Por qué Git es el sistema de control de versiones más popular. [En línea] 12 de mayo de 2014. [Citado el: 8 de diciembre de 2015.] file:///C:/Users/ds/AppData/Roaming/Mozilla/Firefox/Profiles/402230nc.dev-edition-default/ScrapBook/data/20151204164135/index.html.

29. Revelo, James. Hermosa Programación. *Tutorial De Bases De Datos SQLite En Aplicaciones Android.* [En línea] 14 de octubre de 2014. [Citado el: 8 de diciembre de 2015.] http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/.

30. EcuRed. [En línea] 2002. [Citado el: 8 de dicimbre de 2015.] http://www.ecured.cu/Visual_Paradigm. ISBN: 8448132149..

31. Somerville. Materiales_basicos. *eva.uci.cu.* [En línea] [Citado el: 10 de 2 de 2016.] http://eva.uci.cu/file.php/161/Documentos/Materiales_basicos/Materiales_basicos_de_la_Unidad_4/Sommerville/Sommerville_Parte_II_Requerimientos.pdf).

32. —. *Ingenieria de software.* México : s.n., 2011.

33. Ingenieria de Software. *Patrones de diseño.* [En línea] 2011. [Citado el: 14 de marzo de 2016.] http://www.cartagena99.com/recursos/alumnos/apuntes/Patrones%20de%20Diseno.pdf.

34. Aria, Miriela Velazquez. *Componente de medición de la calidad de imágenes de huellas dactilares.* La Habana : s.n., 2014.

35. *Técnicas de pruebas.* 2015.

36. ProcesosdeSoftware. *Metodología XP.* [En línea] 2015. [Citado el: 8 de diciembre de 2015.] https://procesosdesoftware.wikispaces.com/METODOLOGIA+XP.

37. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* Romero, Gladys Marsi Peñalver, Puente, Sergio Jesus García De La y Abad, Abel Meneses. La Habana : s.n., 2010.

38. Ordenadores y Portátiles. *Funciones PDA y como nos pueden ayudar estos dispositivos.* [En línea] 2014. [Citado el: 9 de diciembre de 2015.] http://www.ordenadores-y-portatiles.com/funciones-pda.html.

39. Definición.de. *Definición de perfil.* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] http://definicion.de/perfil/.

40. DefiniciónABC. *Memoria RAM.* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] http://www.definicionabc.com/tecnologia/memoria-ram.php.

41. DefiniciónABC. *Definición de CPU.* [En línea] 2015. [Citado el: 9 de diciembre de 2015.] http://www.definicionabc.com/tecnologia/cpu.php.

42. carlospes.com. *Diccionario de Informática.* [En línea] 2014. [Citado el: 16 de diciembre de 2015.] http://www.carlospes.com/minidiccionario/pseudocodigo.php.

43. Algesa.com.ar. *Diccionario de informática y tecnología.* [En línea] 30 de mayo de 2013. [Citado el: 16 de diciembre de 2015.] http://www.alegsa.com.ar/Dic/api.php.

44. Alegsa.com.ar. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA.* [En línea] 2015. [Citado el: 16 de diciembre de 2015.] http://www.alegsa.com.ar/Dic/script.php.

45. Alegsa, Leandro. ALEGSA.com.ar. *DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA.* [En línea] 12 de mayo de 2010. [Citado el: 15 de dicuembre de 2015.] http://www.alegsa.com.ar/Dic/gps.php.

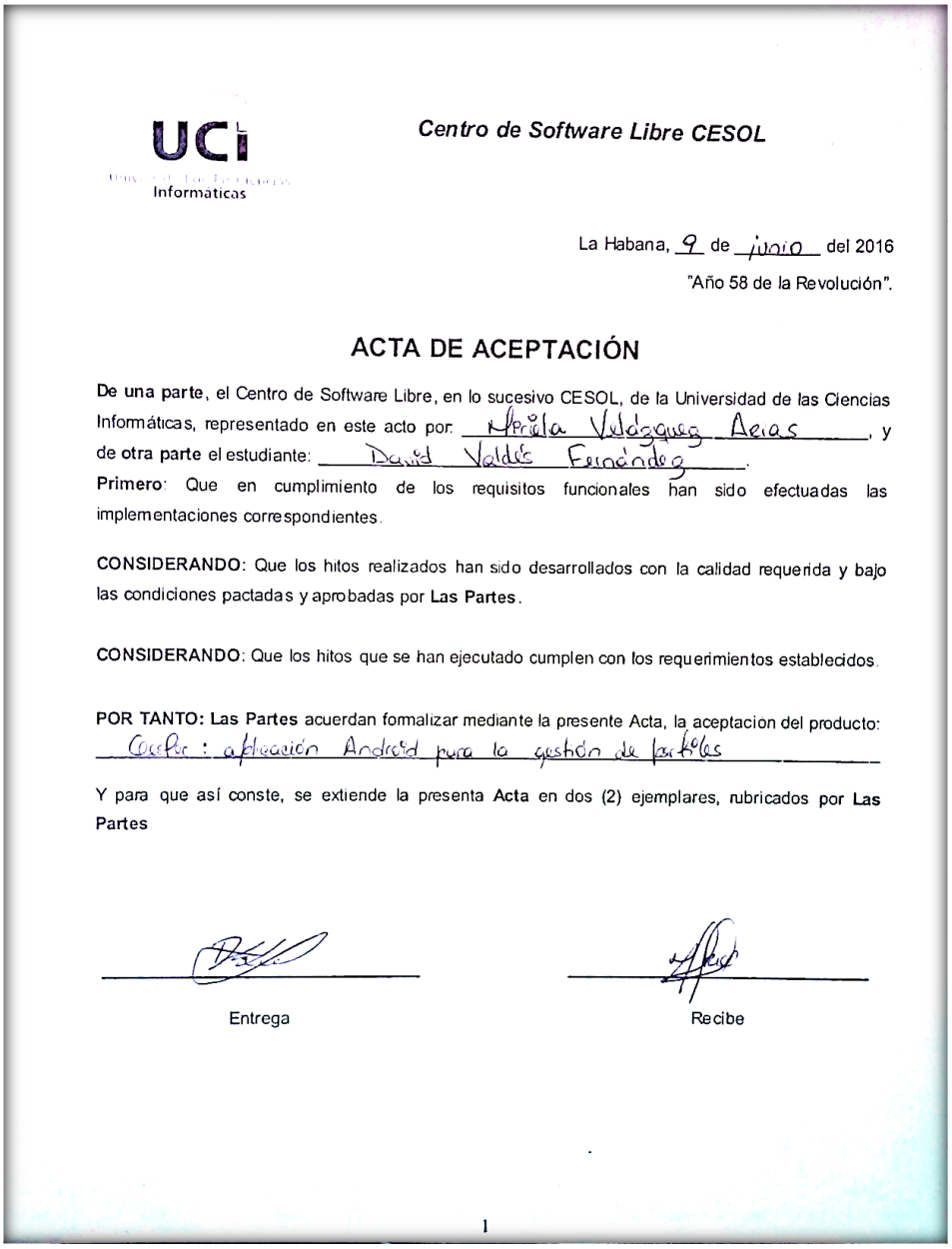
Anexos

Anexo 1: Principales comentarios para las pruebas de aceptación

- Acabo de instalar la aplicación y me parece una solución interesante para administrar el acceso a las aplicaciones ya que en el período de tiempo que desee puedo prescindir de las que quiera. Como recomendación opino que las próximas actualizaciones tengan un diseño más ergonómico para lograr una mejor distribución de sus funciones. Felicidades al programador.
- Muy bien, no he visto la aplicación, pero lo que sería genial es que se integrase con CampusUCI y a través de GPS sepa que estás en un docente, establecer algún perfil de configuración y saber si estás en clases a través del horario o realizando alguna otra actividad.
- Es una buena solución teniendo en cuenta donde vivimos y lo imprescindible que se ha vuelto el uso de *smartphone* en la vida cotidiana. En más de una ocasión las clases, reuniones y demás momentos de seriedad, se han visto interrumpidos por un inoportuno teléfono y su olvidadizo dueño. Además, asumo que contribuya con el ahorro de la batería del dispositivo pues al cancelar ciertos servicios con frecuencia evitará el consumo innecesario de la misma.

Anexo 2: Interfaz de la aplicación GesPer

Anexo 3: Acta de aceptación



UCI
Universidad de las Ciencias Informáticas

Centro de Software Libre CESOL

La Habana, 9 de junio del 2016
"Año 58 de la Revolución".

ACTA DE ACEPTACIÓN

De una parte, el Centro de Software Libre, en lo sucesivo CESOL, de la Universidad de las Ciencias Informáticas, representado en este acto por: Mirla Velázquez Aeras, y **de otra parte** el estudiante: David Valdés Fernández.

Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.

CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requerimientos establecidos.

POR TANTO: Las Partes acuerdan formalizar mediante la presente Acta, la aceptacion del producto: Casefor : aplicación Android para la gestión de parkeos

Y para que así conste, se extiende la presenta **Acta** en dos (2) ejemplares, rubricados por **Las Partes**

Entrega

Recibe

1

Figura 26: Acta de aceptación.

Glosario de términos

- **SO**: Sistema operativo. Conjunto de órdenes y programas que controlan los procesos básicos de un dispositivo y permiten el funcionamiento de otros programas.
- **TIC**: Tecnologías de la Información y las Comunicaciones.
- **PDA**: Asistente Digital Personal.
- **GPS**: Sistema de Posicionamiento Global.
- **CPU**: Unidad Central de Procesamiento.
- **RAM**: Memoria de Acceso Aleatorio.
- **Perfil** de configuración: Grupo de rasgos característicos.
- **DSL**: Lenguaje Específico de Dominio.
- **GRASP**: Patrón de diseño de *software* orientado a objetos.