Facultad 2

Cliente del Gestor de Recursos de Hardware y Software para dispositivos que operan con Android

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

Diplomantes: Yisel Elena Tamayo Betancourt

Rasiel Ernesto Ciervide Cabalé

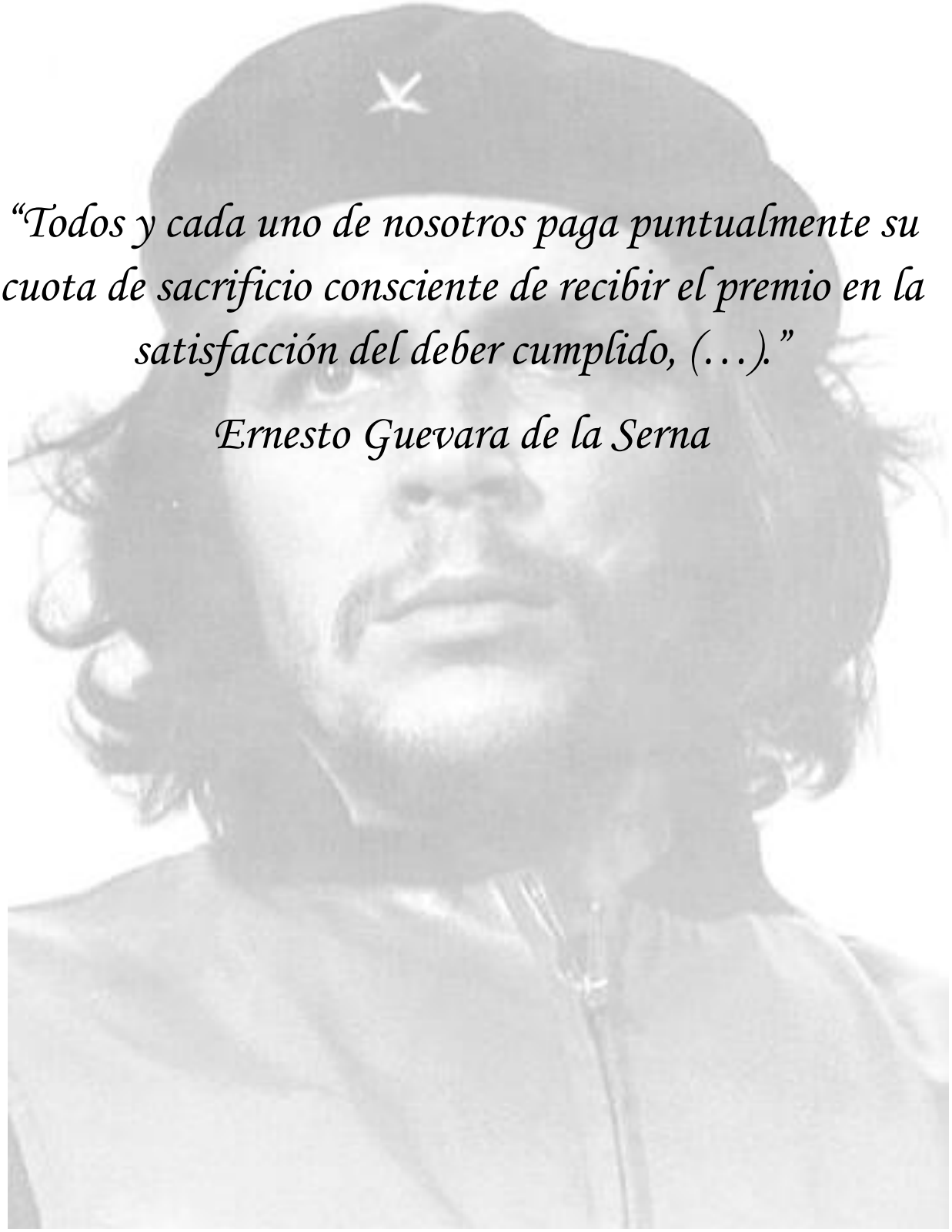
Tutores: Ing. Rainer Segura Peña

Ing. Ernesto Avilés Vázquez

La Habana, junio de 2015

"Año 57 de la Revolución"

Pensamiento



"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, (…)."

Ernesto Guevara de la Serna

Declaración de Autoría

Declaramos ser los únicos autores del trabajo de diploma que tiene como título Cliente del Gestor de Recursos de Hardware y Software para dispositivos que operan con Android y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de junio del año 2015.

Yisel Elena Tamayo Betancourt

_____

Firma del Autor

Rasiel Ernesto Ciervide Cabalé

_____

Firma del Autor

_____

Ing. Rainer Segura Peña

Firma del Tutor

_____

Ing. Ernesto Avilés Vázquez

Firma del Tutor

Dedicatoria

La dedicatoria especial de esta tesis es para ti que ya no estás a mi lado y que de todas las personas de este mundo creo que eres la que más orgullosa se sentía de su Preciosa como te gustaba decirme, a ti que ya no estás, mi abu linda.

A mami por creer en mí, por luchar por mí, por hacer aunque no se podía, por consentirme, por estar ahí para mí todos los días, ahora me toca a mí… Te amo mamá.

A Ernesto Alejandro Tamayo Mesa, porque quiero ser la fuente de inspiración de tu vida, porque dividiste mi cariño y ahí estás, haciendo de mi hermano un hombre mejor.

A Brayam Betancourt Alfonso.

…Yisel Elena Tamayo Betancourt…

Dedico esta tesis a toda mi familia.

A mi mamá por su gran apoyo y por impulsarme siempre a ser mejor, por estar siempre a mi lado, por ser mi amiga, por pedirme siempre hasta lo imposible por ser mejor y estudiar cada día más para ser alguien en la vida, a lo que le estoy y le estaré siempre agradecido.

A mi papá por estar siempre ahí, por creer en mí y apoyarme en todo momento, por conducirme a ser quien soy hoy.

A mis dos bellas hermanas.

A mis abuelas.

A mis tíos y tías.

… Rasiel Ernesto Ciervide Cabalé…

Agradecimientos

De Yisel

Es difícil esto de agradecerles a todas las personas que de una forma u otra me han acompañado en este largo camino, sobre todo a mí porque aunque creía que no podía llegué hasta aquí por mis sueños.

*A mi **Tata** por ser la persona más importante de mi vida, por ser la mitad de mi corazón, **Orquidia** gracias por ser mi mejor amiga, mi madre y mi todo, por darme fuerzas cuando yo creía que no podía, por empujarme un poquito más en el camino de la vida, por ti he llegado hasta aquí y por ti pienso seguir adelante. Gracias por darme razones para probar que una sonrisa tuya me cambia la vida. Te adoro mami, por favor no me faltes nunca. Gracias por ser la mejor madre del mundo, la mía…*

*A mi padre, mi amigo, mi hermano y mi todo **Yuset** gracias por apoyarme siempre, por ser mi fuente de inspiración. Y por estar ahí. A mi papá, que aunque nunca te lo diga te quiero mucho papá. A mi hermano Tito.*

A mi papá de raza Julio, si estuvieras bien yo sé que estarías aquí conmigo.

*A mi tía **Yraida** gracias por ser mi ejemplo de persona en la vida.*

A mi familia aquí en la UCI que estuvieron a mi lado desde el primer año: Yindra, Zenel, Adrián, Yoyi.

A ti por estar en mi vida en los momentos difíciles de segundo y tercer año de la carrera, a pesar de todo te agradezco.

*A mis mejores amigos: **Leanet, Brenda, Jorge Carlos** por ser mi hermanito negro, Arianne, Yasmani, gracias por ayudarme a creer en mí y por estar ahí para compartir las cosas buenas y malas. Al Rafa, Ubel, Annies, Elizabeth, Sevi, Pocho, Osvaldo, Yosbani, Deliannis, Seijas, Alfredo, gracias. A **Ray** por apoyarme desde lejos y darme ánimos, a Marlen por aguantar hasta los últimos momentos.*

A Darlin García Martínez porque cumplí por las dos la promesa de llegar lejos, aunque todavía me falta te quiero mucho mi hermanita.

*A **Rasiel** por aguantarme todo este tiempo mis majaderías, mis regaños, mis acosos, gracias por escogerme como compañera de tesis y por convertirte en un amigo.*

A mis tutores, a Rainer, gracias por ayudarnos y por sacar el tiempo para atendernos, gracias por estar ahí para nosotros, por hacer de este trabajo el suyo propio.

A todos, gracias por ayudarme a ser mejor persona y a convertirme en la profesional que mi mamá siempre esperó de mí, nunca les voy a estar lo suficientemente agradecida.

Agradecimientos

De Rasiel

A toda mi familia.

A mi tío Manolito, a mi primo Adrián, a mi hermanito Daniel, que aunque esté afuera lo sigo queriendo, les estoy especialmente agradecidos a ellos por brindarme los medios para poder realizar esta tesis, que sin ello hubiera sido casi imposible.

A todos los profesores que ayudaron a formarme y que me brindaron su ayuda, a Omar por ser más que un profesor ser un amigo y hacernos pasar tantos buenos ratos en las tardes de fútbol.

A todo el piquete, a Pocho, Pável, Andis, Carlos, Lázaro, el Chino, Paneque con los que he pasado por la mejor experiencia de mi vida sin duda alguna y que llegaron a convertirse en mi segunda familia.

A mi compañera de tesis Yisel, por estar ahí, por tenerme paciencia cuando se estresaba, por convertirse en una amiga.

A mis tutores por ayudarnos y sacar tiempo para nosotros, a Rainer por preocuparse y tomárselo como si fuera su propia evaluación.

A todos los que de una forma u otra hicieron posible que este trabajo fuera posible.

Gracias

Resumen

La Universidad de las Ciencias Informáticas (UCI) como soporte a la industria cubana del software, ha procurado desarrollar un sistema capaz de solventar las carencias existentes respecto al control de inventarios de recursos de hardware y software, denominado Gestor de Recursos de Hardware y Software (GRHS), en el Departamento de Desarrollo de Componentes del centro Telemática (TLM). Actualmente GRHS realiza el control de inventarios para las plataformas de Windows y GNU/Linux.

El desarrollo de un cliente del GRHS para dispositivos que operan con Android, como contribución a la gestión de inventario de hardware y software que se desarrolla en el centro de TLM, constituye la base de la propuesta de solución. Con tal propósito, fue realizado un estudio de los sistemas para la gestión de inventario de recursos de hardware y software concluyendo que ninguno ofrece solución a la totalidad de los objetivos propuestos, asimismo se estudiaron las herramientas empleadas internacionalmente para realizar el reconocimiento y obtención de información en dispositivos que operan con Android.

Palabras claves: Android, dispositivos móviles, gestión de información, GRHS, hardware, software.

Índice

Índice de Tablas

Índice de Figuras

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC´s) han ido evolucionando paulatinamente y revolucionando todos los sectores de la sociedad. Uno de los sectores beneficiados con los desafíos de la nueva era del conocimiento son las telecomunicaciones, como resultado de la búsqueda constante del hombre por satisfacer cada vez más su necesidad de comunicación, logrando la instauración en el mundo de dispositivos cada día más poderosos y veloces en el proceso comunicativo. Del mismo modo y en conjunto con el desarrollo tecnológico de las computadoras tanto portátiles como de escritorio y los satélites ha venido desarrollándose la industria de la tecnología móvil y con esta los sistemas operativos implementados para este tipo de dispositivos.

Un sistema operativo móvil se define como el conjunto de programas cuya misión fundamental es la de gestionar los recursos del dispositivo y en consecuencia constituirá el soporte lógico que controla el funcionamiento del equipo físico. Enfocándose mayormente en la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes formas de manejar la información en ellos. (1)

Entre los más reconocidos pudieran mencionarse IOs, Symbian OS, Blackberry OS, Windows Phone y Android que en los últimos ha entrado con fuerza en el mercado y ha tomado al mundo por sorpresa, siendo considerado el sistema operativo líder en el mercado móvil. (2)

Android es una solución completa de software de código libre para teléfonos y dispositivos móviles. Es distribuido bajo una licencia libre permisiva (Apache) que permite la integración con soluciones de código propietario. Es un paquete que engloba un sistema operativo, un runtime[1] basado en Java, un conjunto de librerías de bajo y medio nivel y un conjunto inicial de aplicaciones destinadas al usuario final.

En el desarrollo de aplicaciones para dispositivos móviles, una de las categorías más importante y con más relevancia es la de sistemas de gestión de información, teniendo en cuenta que la vinculación de la tecnología móvil a los sistemas de gestión de información establece una nueva relación usuario – sistema, que posibilita una fácil socialización de la información propiciando una interacción continua y rápida. Dentro de las telecomunicaciones, los sistemas de gestión de información también marcan una diferencia a la hora de la toma de decisiones, permiten la recolección de datos y la posibilidad de usarlos.

[1] Se denomina **tiempo de ejecución** (runtime en inglés) al intervalo de tiempo en el que un programa de computadora se ejecuta en un sistema operativo.

Sin embargo, aún existen insolvencias en el control de las acciones que se realizan a través de estas tecnologías, o las modificaciones que estas puedan sufrir; se transgrede la privacidad de espacios sociales, institucionales y académicos. Se introducen, en ocasiones, partes duras (tarjetas de memoria, baterías y otros) ajenas e incompatibles con el dispositivo. Dichos problemas abogan de soluciones inmediatas debido al creciente empleo de este tipo de recursos como medios institucionales.

La Universidad de las Ciencias Informáticas (UCI) como institución de la sociedad cubana que sirve de soporte a la industria cubana del software no está exenta de esta necesidad, a raíz de las debilidades confrontadas en el control de estos recursos ha procurado desarrollar un sistema capaz de solucionar dichas carencias, denominado Gestor de Recursos de Hardware y Software (GRHS), implementado en el Departamento de Desarrollo de Aplicaciones del centro TLM.

GRHS es un sistema que funciona a partir de un agente instalado en las PC de la entidad, que envía la información a un gestor, donde se procesa y almacena. Además cuenta con una aplicación web que tributa a la visualización de los datos, la realización de reportes e informes, entre otras características. Esta aplicación permite una mayor facilidad para la gestión de los recursos así como la obtención de informes sobre información tecnológica, evita la duplicación y pérdida de la información, entre otros beneficios. (4)

El sistema aún no es capaz de gestionar la información referente a los dispositivos que operan con Android y que sirven al centro actualmente. Del mismo modo, este elemento ha dificultado la capacidad para controlar todos los recursos que se poseen, ya que resulta muy engorroso descubrir y determinar toda la información que se desea saber de estos dispositivos ya sea que estén defectuosos o que en determinado momento no se encuentran disponibles en el ámbito de trabajo. Surgiendo así la necesidad de crear inventarios de hardware y software para este tipo de recursos. Por consiguiente, disponer de un inventario de estos medios es tener la información, que puede ser vasta y muy detallada, necesaria para la toma de decisiones.

Ejemplo elocuente del tema en cuestión puede ser la información del IMEI[2], fabricante, modelo, procesador y la RAM (Memoria de Acceso Aleatorio) del dispositivo, la lista del software instalada disponible con su versión, así como las características de la pantalla, la unidad de procesamiento gráfico

[2] **IMEI** : **"International Mobile Equipment Identity"**, lo que traducido al español significa **"Identidad Internacional de Equipo Móvil"**, y su propósito es proporcionarle al teléfono celular una identidad única en todo el mundo.

(GPU), cámaras frontales y/o traseras (si las posee), el almacenamiento y existencia de micro SD[3], entre otros de igual relevancia. En tal sentido, la utilidad del inventario será superior en la medida que contenga mayor cantidad de datos, detallados de la mejor manera posible y con la calidad máxima; teniendo en cuenta la relevancia que estos datos puedan representar en el momento de efectuar el inventario. Por ende, para tal finalidad, sería de gran utilidad una herramienta que recoja los datos de un modo automático.

Otra de las problemáticas detectadas es la poca frecuencia con que se realizan los inventarios de estos recursos, hecho que incide en la antigüedad de los mismos, así como en que la información se encuentre en su mayoría descentralizada. Además, el hurto y las violaciones de privilegios y políticas de seguridad son cada vez más frecuentes y con perspectivas de crecimiento, debido a la débil presencia del control.

A lo anteriormente mencionado se unen los siguientes problemas:

- No se tiene control del estado del sistema y el rendimiento, incluyendo la utilización de la unidad central de procesamiento (CPU), la asignación de memoria y utilización.
- Falta de información del estado de los sensores y la batería.
- Falta de información referente a las características de software del dispositivo: versión del sistema operativo o las aplicaciones instaladas.

Pueden añadirse otros, como la posibilidad de ocasionar pérdidas monetarias, ya que la universidad se vería obligada a destinar parte de su presupuesto a adquirir nuevamente estos recursos, gastos cuya influencia afectaría la economía de la institución y, por consiguiente, la del país; ya que podría ser empleado este recurso monetario para la inversión en otro tipo de tecnología cuya utilidad sea más apremiante para el centro.

El uso de este tipo de software especializado en la realización de inventarios para estos dispositivos es una solución relativamente nueva a un problema clásico en la producción: el de controlar y coordinar los materiales para que estén disponibles cuando se precisan.

A partir de todo lo planteado con anterioridad surge como **problema a resolver**: ¿Cómo automatizar el proceso de gestión de información de hardware y software en dispositivos que operan con Android?

[3] **Micro SD:** una **tarjeta SD** (Secure Digital) es una tarjeta de memoria para almacenar contenidos en dispositivos portátiles, como teléfonos móviles, cámaras digitales, tabletas o navegadores GPS.

Para llevar a cabo la investigación se plantea como **objeto de estudio**: los sistemas de gestión de la información de recursos de hardware y software, planteándose como **objetivo general**: desarrollar un cliente para el sistema GRHS que permita automatizar el proceso de gestión de información de hardware y software en dispositivos que operan con Android. La investigación tiene como **campo de acción**: los sistemas para la gestión de recursos de hardware y software en dispositivos que operan con Android.

El objetivo general planteado abarca un conjunto de **tareas de la investigación** para darle cumplimiento, que a continuación se detallan:

- Elaboración del marco teórico de la investigación para definir conceptos fundamentales en la comprensión de la investigación.
- Realización del estudio del estado del arte de los sistemas dedicados a la gestión de la información de hardware y software para establecer semejanzas con las soluciones existentes e identificar características o funcionalidades similares a las del sistema a implementar.
- Análisis de la metodología de desarrollo a utilizar para la descripción de los artefactos que se generan durante la implementación del cliente GRHS para dispositivos que operan con Android.
- Selección de las tecnologías y herramientas necesarias para el desarrollo del cliente GRHS para dispositivos que operan con Android.
- Análisis de las características que debe tener el cliente GRHS para dispositivos que operan con Android para determinar funcionalidades.
- Implementación del Cliente GRHS para dispositivos que operan con Android.
- Estudio de las características del sistema GRHS para lograr la integración del cliente GRHS para dispositivos que operan con Android con el sistema.
- Estudio de los métodos de pruebas existentes para la validación del sistema a partir de pruebas unitarias y de aceptación.

Durante la investigación fueron utilizados como **métodos científicos** los siguientes:

Métodos teóricos:

Histórico - Lógico: posibilitó el estudio analítico de la trayectoria histórica de las herramientas de gestión de inventarios de hardware y facilitó conocer de forma general el funcionamiento y desarrollo de las mismas, permitiendo caracterizar así cada una en sus aspectos más externos.

Analítico – Sintético: este método permitió el análisis de los documentos, procesos y teorías para la extracción de los elementos más importantes que se relacionan con los sistemas de gestión de información de recursos de hardware y software. Además, para buscar y analizar la información acerca de las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema, seleccionando los principales elementos y características.

Modelación: permitió realizar el proceso mediante el cual se crean criterios y alternativas para la propuesta de solución, facilitando la definición de los componentes de la misma así como sus relaciones. Además fue empleado para la realización de los artefactos que requiere la metodología seleccionada: historias de usuarios (HU), tarjetas CRC (Clase, Responsabilidad, Colaborador), Tareas de la Ingeniería, Plan de Iteraciones y Plan de Entregas.

Métodos empíricos:

Entrevista: la utilización de este método permitió la implementación del Cliente GRHS para Android, ya que para la obtención de un servicio de alta calidad se requieren una serie de entrevistas con el cliente y trabajar sobre la base de estas para la satisfacción de sus necesidades. Asimismo se realizaron entrevistas con el cliente para definir los problemas existentes que posteriormente serían resueltos, identificando los recursos que pueden ser controlados. Incluso fueron entrevistados el líder y el analista principal del proyecto GRHS del centro TLM. (Ver modelo de entrevista en el ¡Error! No se encuentra el origen de la referencia..)

Observación: se realizó una observación de los diferentes sistemas existentes a nivel mundial de gestión de recursos de hardware y software específicamente en aquellos destinados a la gestión en dispositivos que operan con Android, y en el caso de Cuba un estudio del proyecto GRHS específicamente del cliente GRHS para sistemas operativos de PC´s como Linux, Debian y Windows que sirvió como modelo para la implementación de las funcionalidades para el cliente GRHS para dispositivos que operan con Android.

Para hacer más comprensible el presente trabajo a continuación se muestra la estructura capitular del mismo:

Capítulo 1. "Marco teórico de la investigación": se realiza un estudio sobre soluciones informáticas similares existentes en sistemas para la gestión de recursos de hardware y software en dispositivos que operan con Android, en Cuba y a escala mundial. Además se analizan los principales elementos teóricos

que constituyen la base de la investigación, entre los cuales se encuentran la metodología de desarrollo para el sistema, las tecnologías y herramientas a utilizar en la propuesta de solución.

Capítulo 2. "Exploración y Planificación": Se aborda el entorno de desarrollo definido, se explican las diferentes características que va a presentar el sistema a implementar y sus funcionalidades siguiendo los aspectos que plantea la metodología ágil XP en sus fases de Exploración y Planificación.

Capítulo 3. "Iteraciones y Producción": Se puntualizan las iteraciones llevadas a cabo durante la etapa de construcción del sistema, definiéndose la arquitectura del mismo, las tarjetas CRC y las tareas de la ingeniería generadas por cada HU. Por último se realizan las pruebas a la propuesta de solución para verificar el cumplimiento de todas las funcionalidades.

Capítulo 1: “Marco Teórico de la Investigación”

Introducción

En el presente capítulo se realiza un estudio de las diferentes aplicaciones informáticas existentes dedicadas a la gestión de recursos de hardware y software de recursos en dispositivos que operan con Android, en distintos países, incluyendo a Cuba. Se enuncian conceptos fundamentales de la gestión de información de recursos de hardware y software. Se describen las características del equipo de trabajo y se exponen las razones de la selección de la metodología de desarrollo. Además son fundamentadas las herramientas y tecnologías que se utilizarán para el desarrollo de la investigación.

1.1 Conceptos fundamentales

A continuación se enuncian conceptos fundamentales para ayudar al entendimiento del proceso de gestión de la información de recursos de hardware y software.

1.1.1 Gestión de Información

La gestión de información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de la organización. Con el objetivo de mostrar cómo influye la gestión de información en el desarrollo y fortalecimiento de las instituciones. Se define como el conjunto de actividades realizadas con el fin de controlar, almacenar y, posteriormente, recuperar adecuadamente la información producida, recibida o retenida por cualquier organización en el desarrollo de sus actividades. (5)

1.1.2 Hardware

El término hardware se refiere a todas las partes físicas de un sistema informático. (1) Son cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado. El término es propio del idioma inglés (literalmente traducido: partes duras); la Real Academia Española lo define como el conjunto de componentes que integran la parte material de una computadora. (6) No solamente se aplica a las computadoras; del mismo modo, también un robot, un dispositivo móvil, una cámara fotográfica, un reproductor multimedia o cualquier otro electrónico que procese datos poseen hardware. El hardware por sí solo no puede hacer nada, pues es necesario que exista el software, que es el conjunto de instrucciones que hacen funcionar al hardware.

1.1.3 Software

El software es un conjunto de programas, documentos, procedimientos, y rutinas asociados con la operación de un sistema de cómputo. Distinguiéndose de los componentes físicos llamados hardware. Comúnmente a los programas de computación se les llama software; el software asegura que el programa o sistema cumpla por completo con sus objetivos, opera con eficiencia, está adecuadamente documentado, y suficientemente sencillo de operar. Es simplemente el conjunto de instrucciones individuales que se le proporciona al microprocesador para que pueda procesar los datos y generar los resultados esperados. (1) Un software puede ser ejecutado por cualquier dispositivo capaz de interpretar y ejecutar las instrucciones para lo cual es creado.

1.1.4 Gestión de Información de Recursos de Hardware y Software

Se define la gestión de información de recursos de hardware y software como el proceso planificación y control de todos los recursos de hardware y software, destinado a facilitar la toma de decisiones disponiendo de información precisa sobre estos.

1.2 Sistemas para la gestión de recursos de hardware y software en dispositivos que operan con Android

Android surge como resultado de la Open Handset Alliance un consorcio de 48 empresas distribuidas por todo el mundo con intereses diversos en la telefonía móvil y un compromiso de comercializar dispositivos móviles con este sistema operativo. El desarrollo viene avalado principalmente por Google (tras la compra de Android Inc. en 2005) y entre las compañías encontramos: de software, operadores, fabricantes de móviles (Motorola, Samsung, LG, HTC y otros) o fabricantes de Hardware. (3)

Las aplicaciones para Android se escriben y desarrollan en Java aunque con unas interfaces de programación de aplicaciones propias o APIS como son conocidas comúnmente, por lo que las aplicaciones escritas en Java para computadoras personales o PC´s (acrónimo de las palabras inglesas Personal Computer) y demás plataformas ya existentes no son compatibles con este sistema. Estas se ejecutan Dalvik, una máquina virtual Java desarrollada por Google específicamente para ser utilizada en los dispositivos que operan con Android con un conjunto de características que la diferencian del resto de las máquinas virtuales Java. (3)

En la actualidad, con la creciente influencia de las tecnologías informáticas, existen varios productos a nivel internacional que se encargan, entre otras funcionalidades, de realizar la gestión de recursos de hardware y software en dispositivos que operan con Android. El uso de estos sistemas ha aumentado de

forma exponencial en los últimos dos años, y como consecuencia, han pasado del total desconocimiento a la práctica cotidiana en el mundo de los negocios, en las universidades y en los organismos gubernamentales.

A continuación se brinda una muestra de soluciones de sistemas que se utilizan para la gestión de recursos de hardware y software; teniendo en cuenta que en el momento de realizar la investigación estas aplicaciones eran las más completas en cuanto a funcionalidades que ofrecían la mayor cantidad de información de recursos de hardware y software.

1.2.1 System Info Droid 1.3.9

System Info Droid es una aplicación que ofrece a los usuarios información en tiempo real, rápida y completa acerca del sistema y del hardware del dispositivo Android (teléfonos, tabletas, portátil Android, entre otros). Permite compartir las características del dispositivo en los programas de mensajería y chat favoritos, así como enviar el informe completo con todos los detalles a la dirección de correo que elija el usuario. Además incorpora un widget[4] inteligente que informa de la temperatura del dispositivo, del uso y la frecuencia del procesador en tiempo real. (7)

Características principales (7)

- Compartir un informe de la información en los programas de mensajería favoritos del usuario y exportar el informe completo mediante correo electrónico.
- Ofrece widget de CPU con la temperatura, la frecuencia y el porcentaje de uso en tiempo real.
- Brinda información del procesador (núcleos, frecuencia máxima, frecuencia en tiempo real).
- Información del dispositivo (versión de Android, kernel, compilación).
- Información del chip gráfico (GPU, vértices, texturas y otros) y del chip de sonido (chip de sonido, modos de sonido, códecs, radio).
- Espacio de almacenamiento y temperatura del dispositivo.
- Información de la pantalla, RAM, sensores y cámaras.
- Es una aplicación que necesita de pocos permisos.

[4] Un ***widget*** o **artilugio** es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o *Widget Engine*. Entre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

1.2.2 Android Assistant

Android Assistant es una poderosa herramienta para la información de navegación en la plataforma Android. Esta aplicación se puede utilizar para ver información de software, información de hardware, información de servicios en ejecución, y la información de la red. También puede ser empleado para ver el sistema de archivos. (8)

Características principales (8)

- Acceder al Android Generic 2.3.3 - 2.3.7
- Vista de software y hardware, explorador para el sistema de archivos.
- Añadir funciones de temporizador para matar a la ejecución de aplicaciones que están al fondo.
- Desinstalar los programas.
- Compartir archivos a través de correo electrónico, bluetooth, Facebook,Twitter y otras.
- Visor de imágenes y visor de vídeos incorporado.

1.2.3 Z-Device Test

Z-Device Test es una aplicación para teléfonos inteligentes (smartphones) que permite comprobar los sensores de dispositivos Android y el estado de los componentes en un análisis intuitivo y completo para ofrecer de manera exhaustiva todas las características de hardware y software. (8)

Características principales (8)

- Complementar información técnica del fabricante.
- Comparar las diferencias técnicas entre más de 600 modelos de dispositivos en el mercado.
- Probar el buen funcionamiento de todos sus sensores así como la exactitud de cada sensor.
- Para obtener el máximo provecho de su teléfono inteligente.
- Comprobar la señal GPS, cámara, brújula, acelerómetro, micrófono, altavoces, señales Wi-Fi, Bluetooth, radio FM, pantalla, batería, CPU, memoria, vibración, USB, Audio/Vídeo, el sistema operativo, sensor de luz, sensor de proximidad, sensor de temperatura, sensor de flash, barómetro y la detección de satélites.

1.2.4 Elixir 2

Es una herramienta que ofrece información detallada sobre el hardware de un dispositivo Android, incluyendo el estado de la batería y los ciclos de carga restante, las redes inalámbricas (3G, Wi-Fi y

Bluetooth), de software, almacenamiento interno y SD, la CPU y la utilización de la memoria, la configuración del sistema, y otros componentes de hardware. (8)

La aplicación inicia al arrancar el dispositivo, y suele ser demasiado intensa en recursos por sí misma porque toca prácticamente todos los recursos del dispositivo Android y requiere permisos graves en el sistema operativo es decir consume demasiados recursos y para ejecutar determinado monitoreo sobre el dispositivo necesita permisos para operar sobre el sistema operativo. Elixir 2 puede ser excesivo para algunas personas que quieren un monitor de sistema, o una herramienta para mantener control en su almacenamiento o procesos en ejecución.

Características principales (8)

- Permite a los usuarios cambiar la configuración del sistema (brillo, tiempo de espera, el volumen, timbre, redes), activar o desactivar los sensores de hardware y administrar aplicaciones instaladas de la pantalla de información dentro de la aplicación.
- Ofrece widgets de la pantalla de inicio que se pueden personalizar para caber en cualquier pantalla de inicio y proporcionar acceso directo a los ajustes del sistema de uso frecuente o simplemente muestran el rendimiento del sistema y la utilización de recursos.
- Ofrece aplicaciones opcionales personales y administración de los complementos que gestionan los contactos, llamadas perdidas, mensajes SMS, y las tareas de nivel de administrador para los usuarios que están dispuestos a otorgar permisos adicionales.

1.2.5 MyDroid System Info

MyDroid System Info es un sistema espectador de información, diseñado para la vigilancia de los recursos del sistema, monitoreo de sensores en tiempo real, registrador y espectador. Es utilizado para verificar la información del dispositivo, identificar y analizar el rendimiento y otras cuestiones, sensores de monitorización. Fue probado en una amplia gama de dispositivos, y que constantemente tratan de mejorar la herramienta para proporcionar información más detallada y precisa. Esto es realmente una gran herramienta desarrollada para los desarrolladores de Android y aficionados por igual. (8)

Características Principales (8)

- Del hardware y sistema operativo: información sobre el dispositivo y sus características (CPU, pantalla, almacenamiento, OpenGL, cámaras).
- Monitor de recursos del sistema (uso de la CPU, utilización de memoria RAM, batería, almacenamiento, red, Wi-Fi).

- Monitoreo de sensores en tiempo real (todos los sensores compatibles con el dispositivo y el sistema operativo Android).
- Visor de registro para ayudar a diagnosticar problemas.

1.2.6 AIDA64 para Android

AIDA64 para Android, es una aplicación para brindar información de hardware y software en dispositivos basados en Android. Desarrollada en abril de 2015 por la compañía FinalWire LTD. En sus paneles, AIDA64 muestra información sobre todos los dispositivos de hardware, programas instalados y rendimiento de los componentes. Basado en el amplio conocimiento del hardware de AIDA64 para Windows, AIDA64 para Android es capaz de mostrar información variada de diagnóstico para teléfonos, tabletas, relojes inteligentes y TV, incluyendo:

- Sistema: Todos los datos generales del sistema, como el modelo del teléfono, las funciones o el almacenamiento.
- CPU: Información técnica sobre el procesador y el nivel de carga.
- Pantalla: Todos los datos referentes a la pantalla: tamaño, resolución, datos sobre la GPU y otros.
- Conexiones: Datos sobre las redes móviles y las redes Wi-Fi a la que se conecta el dispositivo.
- Batería: Información sobre la carga, salud de la batería, tecnología, temperatura, voltaje y capacidad.
- Android: Datos puramente técnicos sobre la versión Android instalada en el dispositivo.
- Dispositivos: Información sobre las cámaras y todos los dispositivos externos conectados.
- Sensores, archivos del sistema y directorios (rutas del sistema para determinadas tareas).
- Datos sobre las aplicaciones instaladas.
- Valores de todos los sensores de temperatura presentes en el dispositivo.
- Los códecs instalados y para qué sirven. (9)

Problemas conocidos (10)

- Cálculo del tamaño de la pantalla diagonal puede ceder el paso a un valor incorrecto si el fabricante ha codificado los valores equivocados en el perfil Android del dispositivo.
- Capacidades de la cámara pueden mostrar información incorrecta si el fabricante ha codificado valores incorrectos en el perfil de Android del dispositivo.

- Capacidad de la batería sólo puede ser reportada por baterías de fábrica. Si la batería se sustituyó por una batería de capacidad extendida, ni Android o AIDA64 serán capaces de detectar la nueva capacidad.
- Tasa de carga de la batería se puede informar incorrectamente si el teléfono o la tableta no admite correctamente las nuevas llamadas a la API baterías introducidas en Android 5.0. Incluso los nuevos dispositivos Android lanzado en 2015, que viene con Android 5.0 y versiones posteriores no admitan correctamente (ejemplo: Galaxy S6 no admite plenamente las nuevas APIs).

1.2.7 Conclusiones parciales

De manera general estas soluciones informáticas brindan información de los dispositivos como: los datos generales del sistema, modelo, almacenamiento, características de la batería, las cámaras, la CPU, el microprocesador, los sensores, la versión de Android, las aplicaciones instaladas y otros.

Constituyen herramientas para gestionar los recursos de hardware y software en dispositivos que operan con Android a nivel de usuario, y si bien de manera general ofrecen una amplia cantidad de información acerca del dispositivo, no permiten mantener un control sobre esta información a nivel institucional, y no facilitan la toma de decisiones respecto al manejo y disponibilidad de estos recursos. No permiten la conexión a la red para controlar desde un servidor cómo son manipulados los recursos de hardware y software en el dispositivo, o ejecutar acciones ante las incidencias detectadas en este.

Actualmente el sistema GRHS realiza la gestión de inventario de todos los recursos referentes a computadoras: laptops o computadoras de escritorio, para las plataformas de Windows y GNU/Linux; sin embargo no ofrece una funcionalidad que permita manejar la información de los recursos de hardware y software para dispositivos que operan bajo la plataforma Android.

1.3 Metodologías de Desarrollo

En la actualidad existen numerosas propuestas de metodologías que inciden de distintas formas en el proceso de desarrollo del software. Estas metodologías de desarrollo son marcos de trabajo que constituyen un conjunto de procedimientos y técnicas usadas para estructurar, planificar y controlar este proceso de desarrollo de software generando la documentación necesaria para el mismo.

Estos marcos de trabajo indican qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener, detallan la información que se debe producir como resultado de una actividad y la

información necesaria para comenzarla. Dentro de estas metodologías existen dos grandes grupos, las metodologías tradicionales o pesadas y las ágiles.

Las metodologías tradicionales se centran fundamentalmente en el control y definición de los procesos, tareas y herramientas a utilizar. Requieren de una extensa documentación ya que pretenden prever todo de antemano y pueden ser muy efectivas para proyectos de gran tamaño. Proponen un gran cúmulo de documentación de acuerdo al tamaño del proyecto y como consecuencia del equipo de desarrollo, es recomendable utilizarla en el desarrollo de proyectos medianos y grandes, por lo que no se consideró su utilización en el desarrollo del presente trabajo.

Por otro lado están las metodologías ágiles que se encargan de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados. Al utilizar este tipo de metodología se hace mucho más significativo crear un producto de software que funcione, en lugar de escribir mucha documentación. Asimismo, se considera más importante la capacidad de respuesta ante un cambio realizado que el seguimiento estricto de un plan.

1.3.1 Metodología a utilizar

Como guía para lograr el proceso de desarrollo de la propuesta de solución se decide emplear la metodología Programación Extrema (XP, eXtreme Programming) la cual se distingue, fundamentalmente, por su pronta, concreta y continua retroalimentación en ciclos cortos de tiempo, así como por su enfoque de planificación incremental, el cual permite definir un plan general que se espera evolucione durante la vida del proyecto. (11) XP posee también facilidad para flexibilizar la implementación de funcionalidades, respondiendo a los cambios que el negocio necesita. Se encuentra bien documentada con innumerables recursos en línea disponibles, comunidades libres y grupos de noticias, encontrándose una gran cantidad de proyectos desarrollados con esta metodología.

La metodología XP ostenta de una gran dependencia de las pruebas automatizadas escritas por los propios programadores y de la supervisión por parte del cliente en el proceso de desarrollo para permitir la evolución del sistema y capturar los defectos antes de finalizar cada iteración. Además manifiesta la dependencia de un proceso de diseño evolutivo que dura todo el tiempo en que se desarrolla el sistema y su estrecha y cercana colaboración de los programadores con habilidades ordinarias. (12)

Se adapta perfectamente a las características del equipo de desarrollo y del proyecto. Asimismo, se caracteriza por centrarse en las necesidades del cliente, siguiendo una serie de reglas para lograr un producto de buena calidad en poco tiempo, proponiendo iteraciones muy cortas. (11) Esta metodología es

también muy efectiva cuando se exige reducir los tiempos de desarrollo mientras mantiene una alta calidad.

Tabla 1: Características del equipo de desarrollo.

Características	Valoración
Composición del equipo de trabajo	4 Personas ➢ 2 clientes ➢ 2 desarrolladores
Experiencia productiva	(alta, media, baja) ➢ Media
Conocimientos de metodologías ágiles	(bajo, medio, avanzado) ➢ cliente (medio)
Conocimientos de metodologías tradicionales	(bajo, medio, avanzado) ➢ cliente (medio) ➢ desarrolladores(bajo)
Conocimientos del Lenguaje de Programación Java	(bajo, medio, avanzado) ➢ cliente (bajo) ➢ desarrolladores(medio)

A continuación se muestran las razones de la selección:

- El proyecto es pequeño y todo el trabajo es llevado a cabo por una pareja de programadores.
- La generación de artefactos y roles en exceso no es necesaria ya que el proyecto es pequeño y está diseñado para ser realizado en el menor tiempo posible.
- Los requisitos suelen ser cambiados con frecuencia en la medida en que avanza el desarrollo del proyecto, así el cliente puede ir añadiendo HU, dividirlas para agilizar el trabajo o eliminarlas simplemente. Esta metodología permite al equipo de trabajo modificar todos los planes conforme a lo anterior.
- Tanto cliente como desarrolladores forman parte del equipo de trabajo de forma tal que se logra una integración cliente - equipo de desarrollo, permitiendo la retroalimentación, corrección de errores y finalmente la realización de un producto capaz de satisfacer todas las necesidades del mismo.

- Se tiene la ventaja de desarrollar pequeñas partes del producto y probarlo, permitiendo terminar todas las funcionalidades e integrar el producto final.

1.4 Técnicas y herramientas

A fin de garantizar a lo largo de todo el proceso de desarrollo, que el sistema cumpla con los requisitos tanto funcionales como no funcionales que fueron identificados, se utilizaron un conjunto de técnicas, herramientas y lenguaje de programación, los cuales se describen a continuación.

1.4.1 Lenguaje de Programación

Un lenguaje de programación es un lenguaje formal que permite controlar el comportamiento físico y lógico de un ordenador mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos de hardware y software existentes. (13)

Java

Java es un lenguaje de programación orientado a objetos, el cual agrupa en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. Diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. (14) El código binario de Java no es un lenguaje de alto nivel, sino un verdadero código máquina de bajo nivel, viable incluso como lenguaje de entrada para un microprocesador físico.
Se determinó utilizar Java 8 para la implementación del cliente GRHS para dispositivos que operan con Android teniendo en cuenta que es un lenguaje orientado a objetos de gran potencialidad, robustez, seguro y multiplataforma, el equipo de desarrollo tiene conocimientos sobre este lenguaje y posee experiencias, capacidades y habilidades con el desarrollo de aplicaciones en Java. Además porque es el lenguaje nativo en el desarrollo de aplicaciones Android.

1.4.2 Entorno integrado de desarrollo (IDE)

Un IDE es un programa que comprende un entorno de programación amigable para uno o varios lenguajes, brindando facilidades al programador, tales como: un editor de código, un compilador, un depurador, y opcionalmente, un constructor de interfaz gráfica.

Eclipse 4.2.1

Eclipse es un entorno de desarrollo integrado de código abierto, multiplataforma, cuenta con capacidad de completamiento de código, ambiente flexible y rápido. Es un producto singular por la versatilidad que ofrece, y todas estas características están determinadas por la arquitectura con la cual fue diseñado. Eclipse es considerado un producto inigualable por su arquitectura madura, bien diseñada y extensible basada en plugins[5]. (15) Sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como son C/C++, PHP o Python. La arquitectura de plugins de Eclipse permite, además de integrar diversos lenguajes sobre el mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías y otras. (16)

Complementando lo señalado el equipo de desarrollo posee la experiencia, los conocimientos, habilidades y capacidades en el trabajo con este IDE por lo que se decidió utilizarlo para desarrollar la propuesta de solución siguiendo además las políticas de seguridad que se rige la Universidad por ser un IDE que se comercializa bajo una licencia de software libre. Otro factor fundamental para la decisión es el tiempo limitado que se dispone para el desarrollo del sistema lo que dificulta la posibilidad de seleccionar un IDE al cual se desconoce por completo.

Android Developer Tools

Para desarrollar la propuesta de solución se empleará ADT (Android Developer Tools) que es un plugin para Eclipse que proporciona un conjunto de herramientas que se integran con este IDE. Ofrece acceso a muchas características y un potente entorno integrado que permite desarrollar aplicaciones para Android. (17) Extiende las capacidades de Eclipse para que pueda configurar rápidamente nuevos proyectos para Android, construir una interfaz de usuario de aplicación, depurar su aplicación, y la exportación firmado (o no firmados) de paquetes de aplicaciones para su distribución.

Software Development Kit (SDK) 21

SDK en español Kit de Desarrollo de Software es generalmente un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, por ejemplo

[5] Un **plugin** es un programa que incrementa o aumenta las funcionalidades de un programa principal. Por lo general es producido por una compañía diferente a la que produjo el primer programa.

ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, entre otros.

El SDK es una API creada para permitir el uso de cierto lenguaje de programación, o puede, también, incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas más comunes incluyen soporte para la detección de errores de programación IDE y otras utilidades. Los SDK frecuentemente incluyen códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material de referencia primario. (17)

El empleo del SDK 21 fue necesario en el desarrollo del sistema para la integración del IDE con los plugins correspondientes para el desarrollo de aplicaciones en Android.

1.4.3 JavaScript Object Notation (JSON)

JSON es un formato ligero para intercambio de datos que surge como alternativa a XML en AJAX[6]. Se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia, cuando la fuente de datos es explícitamente de confianza y donde no es importante el no disponer de procesamiento XSLT[7] para manipular los datos en el cliente.

Este formato de datos es más liviano que XML8 y es en apariencia más sencillo, ya que utiliza el código JavaScript9 como modelo de datos. (19)Tiene un formato sencillo y fácil de utilizar tanto para programadores para su lectura y escritura como para máquinas en su análisis y generación. Debido a la simplicidad, no es necesario que se construyan parsers[10] personalizados. Sirve para representar objetos en el lado del cliente, normalmente en aplicaciones RIA (Rich Internet Aplications) que utilizan JavaScript.

Características principales (19)

- Independiente de un lenguaje específico.
- Basado en texto, de formato ligero, fácil de parsear.
- No define funciones.
- No tiene estructuras invisibles, espacios de nombre, validador y no es extensible.

[6] **AJAX**: Acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).
[7] **XSLT:** Lenguaje Extensible de Hojas de Estilo Transformado.
[8] **XML:** Extensible Markup Language en español Lenguaje de Marcas Extensible.
[9] **Javascript:** Lenguaje de programación interpretado o que no requiere compilación.
[10] **Parser:** es un analizador sintáctico constituye una de las partes de un compilador. Es la parte encargada de transformar su entrada en un árbol de derivación.

Durante el desarrollo de la aplicación fue seleccionado como formato para el intercambio de datos entre el cliente GRHS para dispositivos que operan con Android y el servidor GRHS del centro TLM.

1.4.4 Sistemas gestores de base de datos

De forma sencilla un Sistema de Gestión de Bases de Datos (SGBD) se puede definir como un conjunto de programas dedicados para acceder a una colección de datos muy bien interrelacionadas. “Los SGBD, son aplicaciones que permiten a los usuarios definir, crear y mantener la base de datos y proporcionan un acceso controlado a la misma”. (20)

SQLite

SQLite es un manejador de código abierto de bases de datos que combina una interfaz muy limpia de SQL. Permite trabajar con poca memoria y con una velocidad bastante rápida, características que son necesarias cuando se habla de entornos móviles.
SQLite soporta las características estándar de las bases de datos relacionales como la sintaxis que se basa en SQL, transacciones y la elaboración de consultas. Debido a esto, cualquier desarrollador que haya trabajado con bases de datos sin importar el entorno, no encontrará una dificultad especial en trabajar con bases de datos locales en Android.
Las principales características de este gestor de bases de datos son las siguientes (21):

- **Tamaño:** tiene pequeña memoria y una necesita de una biblioteca única para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas.
- **Portabilidad:** se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.
- **Interfaces:** cuenta con diferentes interfaces del API, las cuales permiten trabajar con C++, PHP, Perl, Python, Ruby y Groovy.
- **Costo:** es de dominio público, y por tanto, es libre de utilizar para cualquier propósito sin costo y se puede redistribuir libremente.
- **No posee configuración:** de la forma en que fue creado y diseñado, no necesita ser instalado. No es necesario prender, reiniciar o apagar un servidor, e incluso configurarlo; esta cualidad permite que no haya un administrador de base de datos para crear las tablas, vistas, asignar permisos o bien la adopción de medidas de recuperación de servidor por cada caída del sistema.

En aditamento a las características señaladas, SQLite es el SGBD empleado por los clientes GRHS, por lo que se decide utilizarlo en su versión 3.0 para almacenar los datos generados después de realizado cada inventario.

1.4.5 Lenguaje de modelado

Para graficar las entidades se empleará Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), es un lenguaje utilizado para especificar, visualizar, construir y documentar artefactos de un sistema de software. Está constituido por un conjunto de diagramas y es necesario contar con todos esos diagramas dado que cada uno se dirige a cada tipo de persona implicada en el sistema. Un modelo UML indica qué es lo que supuestamente hará el sistema, mas no como lo hará. (22)

UML proporciona el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. Por las características antes mencionadas y además porque el equipo de desarrollo cuenta con experiencia en el desarrollo de este lenguaje se decide emplearlo en su versión 2.1.

Para efectuar el modelado de los procesos de negocio se empleó BPMN (Notación para el Modelado de Procesos de Negocio), la cual es una notación gráfica estandarizada para el modelado de procesos de negocio. BPMN sirve como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación.

1.4.6 Herramienta CASE

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) son empleadas con el fin de automatizar los aspectos claves de todo el proceso de desarrollo de software de un sistema, desde su inicio, hasta completar todo el proceso de implementación.

Para el desarrollo de la propuesta de solución se empleará Visual Paradigm for UML, una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: negocio, descripción de los requisitos, análisis y diseño orientados a objetos, implementación, pruebas y despliegue. Las principales características de Visual Paradigm for UML son: (23)

- Entorno de creación de diagramas UML 2.1.
- Lenguaje estándar, común a todo el equipo de desarrollo.
- Existen varias versiones compatibles con diferentes plataformas, usadas para necesidades específicas del equipo de desarrollo.
- Permite la integración con Eclipse.

Como demuestran las características antes mencionadas la herramienta Visual Paradigm for UML en su versión 8.0 es un software potente por lo que es utilizada a nivel mundial. Posibilita la realización satisfactoria de los diferentes diagramas en dependencia del flujo de trabajo de la metodología que se utiliza. Por estas razones es la herramienta seleccionada para confeccionar el modelo de datos, el diagrama de despliegue y los diagramas de procesos del negocio.

Conclusiones del capítulo

El desarrollo de un cliente del GRHS para dispositivos que operan con Android, como contribución a la gestión de inventario de hardware y software que se desarrolla en el centro de TLM, constituye la base de la propuesta de solución. Con tal designio, fue realizado un estudio de los sistemas para la gestión de inventario de recursos de hardware y software en dispositivos que operan con Android concluyendo que ninguno ofrece solución a la totalidad de los objetivos propuestos; y de las herramientas y técnicas empleadas internacionalmente para realizar el reconocimiento y obtención de información en dispositivos que operan con Android.

Se definieron las herramientas y tecnologías mayormente adecuadas para dar solución al problema planteado, teniendo en cuenta las características de estas y las necesidades descritas con anterioridad a las que da respuesta la propuesta de solución. Se seleccionó Eclipse 2.4.1 como IDE de desarrollo, Java 8 como lenguaje de programación, UML 2.1 como lenguaje de modelado, Visual Paradigm 8.0 como herramienta para el modelado y SQLite 3.0 como sistema gestor de base de datos. Además, se investigó sobre las metodologías de desarrollo de software, escogiéndose XP como la metodología de desarrollo a utilizar.

Capítulo 2 : “Exploración y Planificación”

Introducción

En este capítulo se aborda el entorno de desarrollo definido, se explican las diferentes características que va a presentar el cliente GRHS para dispositivos que operan con Android a implementar y sus funcionalidades. Para ello se siguen los aspectos que plantea la metodología ágil XP en sus fases de Exploración y Planificación, la cual servirá de guía para tener mayor organización en la distribución del tiempo, actividades y artefactos a desarrollar, confeccionándose las HU que proporcionan un mayor entendimiento y comprensión del sistema y se tratan los principales artefactos generados y la planificación del tiempo y el esfuerzo de las fases posteriores.

2.1. Diseño de la propuesta de solución

La solución propuesta está diseñada para aquellas entidades que hagan uso de dispositivos que operan con Android como recurso institucional. Su objetivo es, una vez integrado al servidor GRHS, contribuir a la gestión de inventario que este realiza monitorizando cada uno de los componentes que posee el dispositivo, detectando modificaciones en estos y respondiendo ante las incidencias detectadas. De igual manera, será capaz de mostrar toda la información de hardware y software que posee el dispositivo Android.

2.1.1 Modelo del Negocio

Por una parte el cliente del GRHS para dispositivos que operan con Android será capaz de recolectar toda la información de hardware del dispositivo tal como IMEI, información de la pantalla, sensores, cámaras, batería, GPU, CPU, datos de fabricación permitiendo verificar con el servidor, si existe un inventario previo qué cambios han ocurrido, detectarlos y ejecutar acciones ante estos, y en caso tal de que no existan incidencias o no existe un inventario inicial enviar el nuevo inventario con las características de hardware del dispositivo. Además de enviar trazas cuando no han ocurrido cambios a partir del último inventario registrado en el servidor, estas trazas serán los datos de fecha y hora en la que es recolectado el inventario.

Por otro lado la aplicación también será capaz de recolectar toda la información correspondiente a las características de software del dispositivo, versión de Android instalada, una lista con las aplicaciones instaladas; ofreciendo así detalles importantes de estas que puedan ser objeto de modificaciones. Además el cliente permitirá al usuario una sucesión de configuraciones que le faciliten la conexión al servidor

GRHS indistintamente. Los procesos descritos se muestran en las imágenes a continuación (ver **Figura 1**, **Figura 2)**.

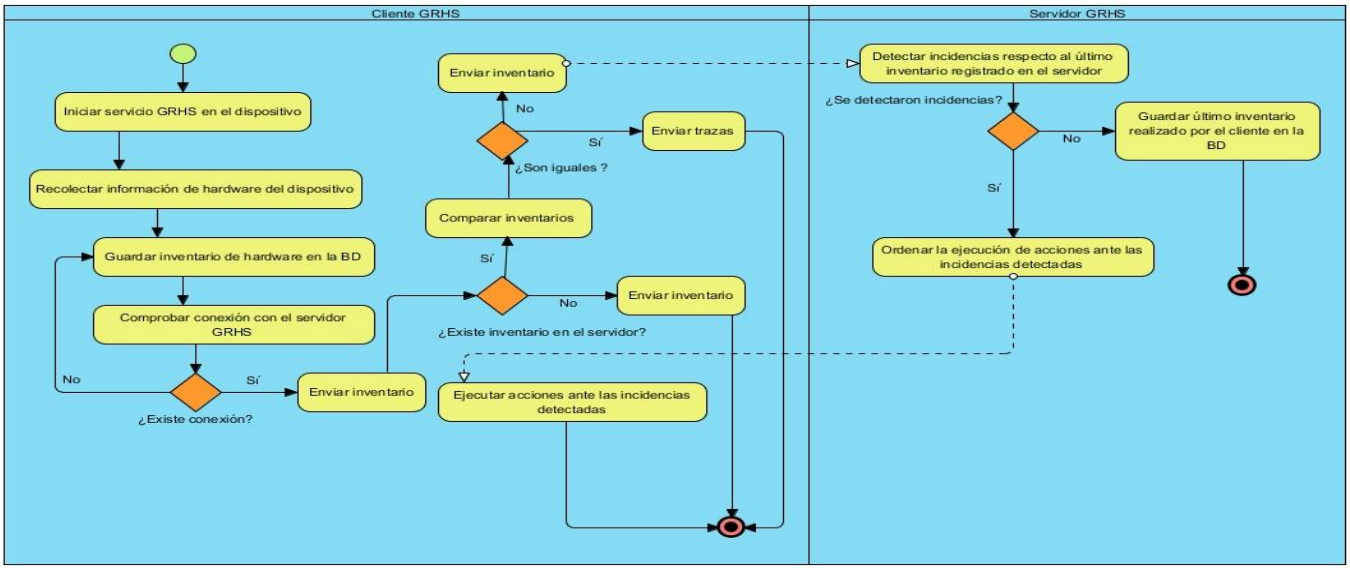


Figura 1: Proceso de Negocio Recolectar información de hardware.

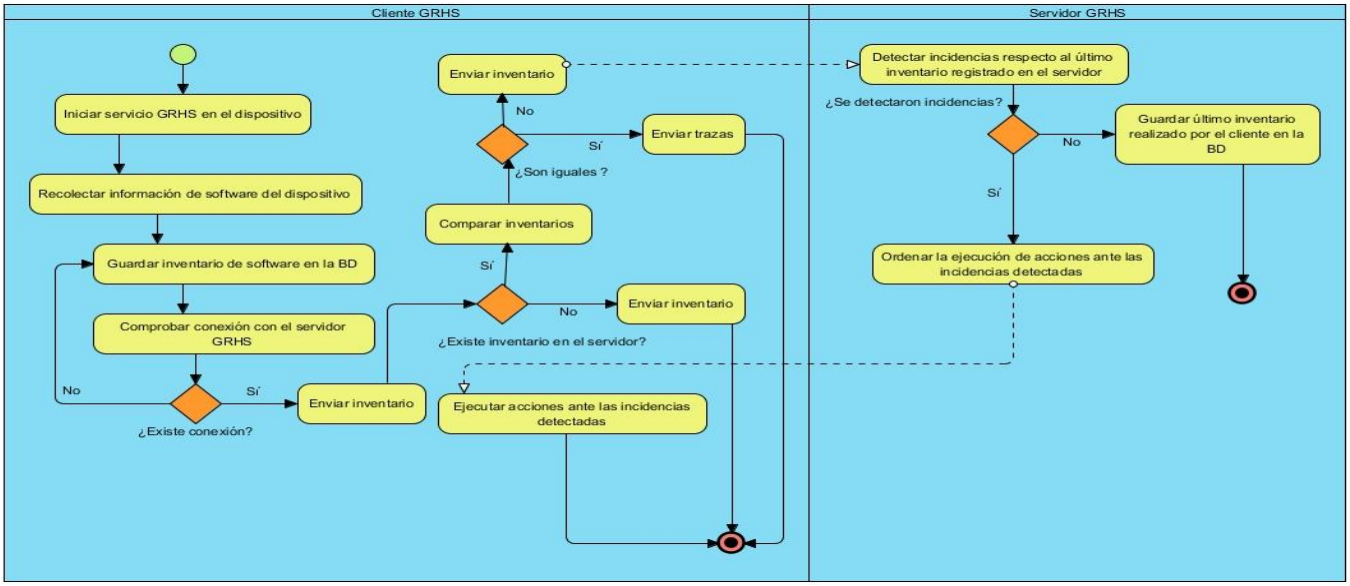


Figura 2: Proceso de Negocio Recolectar información de software.

2.2 Funcionalidades del Cliente GRHS para dispositivos que operan con Android

El análisis de requisitos es la primera fase técnica del proceso de ingeniería de software. La tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado y especificación que

permite definir la función y el rendimiento del software, e indica la interfaz y las restricciones que debe tener el software. (24) La importancia de esta fase radica en la especificidad con que son detallados los requisitos que permitirán el diseño y posteriormente la implementación del sistema. A través de ellos se define el alcance del sistema en cuanto a las acciones que debe realizar, y en cuanto a la transferencia de datos entre todas las diferentes funciones del sistema.

Los requisitos capturados para darle solución al problema y que representan las funcionalidades del sistema que debe cumplir el cliente GRHS para dispositivos que operan con Android son los siguientes:

1. **Realizar inventario del hardware del dispositivo:** consiste en obtener la información de hardware del dispositivo (IMEI, versión del kernel, número de compilación, fabricante, marca, modelo y MAC de la Wi-Fi).
2. **Realizar inventario del CPU del dispositivo:** consiste en obtener información de las características del CPU que posee el dispositivo (arquitectura, procesador, revisión del CPU, hardware del CPU, frecuencia máxima y actual del CPU).
3. **Realizar inventario del GPU:** consiste en obtener información de las características del GPU que posee el dispositivo (renderer, versión y proveedor).
4. **Realizar inventario de la RAM:** consiste en obtener información de la memoria total del dispositivo.
5. **Realizar inventario del almacenamiento del dispositivo:** consiste en obtener información del almacenamiento interno del dispositivo (capacidad de almacenamiento del sistema, capacidad libre de almacenamiento del sistema, capacidad de almacenamiento interno emulado si tiene, capacidad libre de almacenamiento interno emulado si tiene) y el almacenamiento externo (si tiene tarjeta micro SD y en caso de tener obtener la capacidad de almacenamiento, capacidad libre de almacenamiento).
6. **Realizar inventario de la pantalla:** consiste en obtener información de las características de la pantalla del dispositivo: tamaño horizontal y vertical en píxeles, densidad DPI[11] de la pantalla.
7. **Realizar inventario de las cámaras:** consiste en obtener información de las cámaras del dispositivo: cámara frontal (presencia de la cámara frontal, resolución de la cámara en píxeles y en

[11] **DPI:** Los puntos por pulgada (ppp) del inglés *dots per inch* (dpi) es una unidad de medida para resoluciones de impresión, concretamente, el número de puntos individuales de tinta que una impresora o tóner puede producir en un espacio lineal de una pulgada.

megapíxeles) y cámara trasera (presencia de la cámara trasera, resolución de la cámara en píxeles y en megapíxeles).

8. **Realizar inventario de la batería:** consiste en obtener información de la batería del dispositivo: tecnología de la batería, salud de la batería, estado de la batería y nivel de la batería.
9. **Realizar inventario de los sensores:** permite la obtención de una lista que muestra los sensores presentes en el dispositivo.
10. **Realizar inventario del software:** consiste en obtener la información de software del dispositivo: versión de Android, versión del SDK, versión del software, tipo de red telefónica, número del teléfono, operador de la línea, estado de la línea.
11. **Realizar inventario de las aplicaciones instaladas:** permite mostrar una lista de todas las aplicaciones instaladas en el dispositivo, incluye el nombre del paquete de la aplicación.
12. **Enviar inventario:** consiste en realizar un envío al servidor de la información de hardware y software del dispositivo en caso de que exista algún cambio en el inventario o en caso de que no exista inventario del dispositivo en el servidor.
13. **Comprobar inventario:** permite comparar el inventario realizado en el dispositivo con el último inventario registrado en el servidor.
14. **Enviar trazas:** permite enviar al servidor los datos de fecha y hora en que se realizó el inventario en caso de que no se detecten cambios desde el último inventario registrado en el servidor.
15. **Notificar incidencias:** permite notificar al usuario una acción ante una incidencia detectada por el servidor en el dispositivo, consiste en el envío de una notificación al administrador del servidor para que este ejecute o no una acción ante una incidencia detectada.
16. **Configurar dirección del servidor:** permite al usuario local del dispositivo configurar de forma manual la dirección IP y los puertos a través de los cuales enviará el inventario, recibirá configuraciones, acciones ante incidencias y notificaciones del servidor.

2.3 Requisitos del cliente del GRHS para Android

Para asegurar un correcto funcionamiento del sistema deben tenerse en cuenta los siguientes requisitos:

2.3.1 Usabilidad

El usuario necesitará una preparación previa para operar con la aplicación. El manejo de esta es sencillo, permitiendo la fácil comprensión por parte del usuario (en este caso se refiere al usuario que tiene

instalado el GClient en el dispositivo) pero requiere un nivel medio o alto de conocimientos de computación. Deberá saber conectarse a una red Wi-Fi, y ser capaz de configurar en la aplicación los datos del servidor (IP y puerto) GRHS al que debe conectarse el cliente para enviar el inventario.

2.3.2 Disponibilidad

El dispositivo deberá tener acceso a una red Wi-Fi, de manera que el servicio de la aplicación pueda ejecutarse con la mayor frecuencia posible para lograr un grado máximo de actualización de la base de datos del sistema GRHS en tiempo real. Este constituye un elemento fundamental debido a la obtención constante de datos relevantes para el control de los recursos y la toma de decisiones.

2.3.3 Hardware

Para la instalación del cliente debe disponerse de un dispositivo con conexión Wi-Fi, al menos 30 Mb de espacio libre de memoria, memoria RAM de no menos de 512 Mb y un microprocesador con una frecuencia de 800 MHz o mayor.

2.3.4 Software

Para la instalación del cliente debe disponerse de un dispositivo con sistema operativo Android con versión 2.2 o superior.

2.3.5 Interfaz de Usuario

La aplicación propuesta posee una interfaz sencilla destinada a los usuarios que tengan instalado el cliente en el dispositivo. Consta de tres vistas, la primera e inicial mostrará al usuario una lista con los componentes del dispositivo, permitiendo seleccionar uno de estos, la segunda se mostrará una vez que el usuario haya seleccionado un componente y mostrará los datos relacionados a este componente, por último la tercera vista mostrará información acerca de la aplicación. En las dos primeras vistas el usuario tendrá la opción de configurar la conexión con el servidor GRHS.

2.4 Estándares de codificación

La metodología XP promueve la programación basada en estándares, de forma tal que todo el equipo pueda comprenderla y se facilite la recodificación. Los estándares de codificación son pautas de programación que no se encuentran enfocadas a la lógica del programa sino a la estructura y la apariencia física del mismo facilitando la lectura. Mejoran la legibilidad del código al mismo tiempo que permiten su

comprensión rápida. Permite que cualquier programador entienda y pueda mantener la aplicación. En muy raras ocasiones una misma aplicación es mantenida por su autor original.

El establecimiento de los estándares o convenciones de programación empleados en el desarrollo de software sobre la plataforma Java está basado en los estándares recomendados por Sun Microsystems, que han sido difundidos y aceptados ampliamente por toda la comunidad Java, y que han terminado por consolidarse como un modelo estándar de programación (25).

A continuación se muestran las reglas y recomendaciones a seguir para que el código del cliente GRHS para dispositivos que operan con Android sea comprendido por los desarrolladores del sistema GRHS del centro TLM.

2.4.1 Comentarios

Todo fichero fuente debe comenzar con un comentario que incluya el nombre de la clase, información sobre la versión del código, la fecha y el copyright. El copyright indica la propiedad legal del código, el ámbito de distribución, el uso para el que fue desarrollado y su modificación.

Dentro de estos comentarios iniciales podrían incluirse adicionalmente comentarios sobre los cambios efectuados sobre dicho fichero (mejoras, incidencias, errores, y otros). Estos comentarios son opcionales si los ficheros están bajo un sistema de control de versiones bien documentado, en caso contrario se recomienda su uso. Estos comentarios constituyen el historial de cambios del fichero. Este historial es único para cada fichero y permitirá conocer rápidamente el estado y la evolución que ha tenido el fichero desde su origen. La figura 3 muestra un ejemplo de comentario de inicio aplicado en la implementación.

```
/*
 * ClientService.java
 *
 * Version 1.0     Mayo - 2015
 *
 * Centro Telematica
 * Universidad de las Ciencias Informaticas
 */
```

Figura 3: Ejemplo de estándar de comentario.

2.4.2 Declaración de variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas. Las variables nunca podrán comenzar con el carácter "_" o "$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad

la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales (consultar figura 4 para ver ejemplo de estándar de declaración de variables).

```
NotificationManager notificationManager;
PackageManager packageManager;
List<ApplicationInfo> appList;
String collectedInventory;
SharedPreferences preferences;
String ip;
String port;
boolean ipReachable;
List<NameValuePair> parameters;
String fullIP;
boolean serverHaveDb;
```

Figura 4: Ejemplo de estándar de declaración de variables.

2.4.3 Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas como se muestra en la figura 5.

```
public String getMacAddress() {
    @SuppressWarnings("static-access")
    WifiManager wifiManager = (WifiManager) context
            .getSystemService(context.WIFI_SERVICE);
    String address = "";
```

Figura 5: Ejemplo de estándar de métodos.

2.4.4 Sentencias

Las sentencias pertenecientes a un bloque de código estarán tabuladas un nivel más a la derecha con respecto a la sentencia que las contiene. El carácter inicio de bloque "{" debe situarse al final de la línea que inicia el bloque. El carácter final de bloque "}" debe situarse en una nueva línea tras la última línea del bloque y alineada con respecto al primer carácter de dicho bloque. Todas las sentencias de un bloque deben encerrarse entre llaves "{...}", aunque el bloque conste de una única sentencia. Esta práctica permite añadir código sin cometer errores accidentalmente al olvidar añadir las llaves. En las figuras 6 y 7 se muestran ejemplos de estándar de declaración de sentencias.

```
if (wifiManager.isWifiEnabled()) {
    // WIFI ya esta habilitada.
    WifiInfo wifiInfo = wifiManager.getConnectionInfo();
    address = wifiInfo.getMacAddress();
} else {
    // Habilita la WIFI primero
    wifiManager.setWifiEnabled(true);
    // WIFI esta ahora habilitada.
    WifiInfo wifiInfo = wifiManager.getConnectionInfo();
    address = wifiInfo.getMacAddress();
    wifiManager.setWifiEnabled(false);
}
```

Figura 6: Ejemplo de sentencia if- then-else.

```
try {
    String[] args = { "/system/bin/cat", "/proc/cpuinfo" };
    result = cmdexe.run(args, "/system/bin/");
    Log.i("result", "result=" + result);
} catch (IOException ex) {
    ex.printStackTrace();
}
```

Figura 7: Ejemplo de sentencia try-catch.

2.5 Fase de exploración

La fase de exploración es la primera fase definida por la metodología XP, en esta se define el alcance real del sistema permitiendo una familiarización del equipo de desarrollo con las herramientas, tecnologías y procesos. Esta fase comienza por la creación de una serie de historias, llamadas HU las cuales definen mediante su redacción qué es lo que verdaderamente necesita el cliente y es aquí donde los programadores estiman el tiempo de desarrollo. (12)

2.5.1 Historias de Usuarios

Las HU son la técnica utilizada en XP para especificar las funcionalidades del sistema, brindan detalles sobre la estimación del riesgo y cuánto tiempo será empleado en su implementación. El cliente es el encargado de asignarle una prioridad a cada HU y es el equipo de desarrollo el encargado de asignarle un costo, este costo se traduce en las semanas que llevará el desarrollo de las mismas. Si las HU según lo planificado demoran en desarrollarse, se sugiere dividirla en HU más pequeñas. Es importante destacar que las HU nuevas pueden describirse en cualquier momento, con esto se comprueba la flexibilidad de la metodología.

La prioridad en el negocio.

Alta: Son aquellas HU que constituyen funcionalidades fundamentales en el desarrollo el sistema, a las que el cliente define como principales para el control integral del sistema.

Media: Son las funcionalidades a tener en cuenta por el cliente, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: Es otorgada a las HU que son funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo.

Para hacer más comprensible las HU, a continuación el equipo de desarrollo enuncia su leyenda:

Número: número de identificación para las HU, sería incremental en el tiempo.
Nombre Historia de Usuario: es el nombre de la HU, sirve para identificarla fácilmente tanto para los desarrolladores como para los clientes.
Modificación de Historia de Usuario Número: cantidad de modificaciones que se le ha realizado a la HU (de no tener modificaciones se pone ninguna, si no la cantidad de veces que ha sido modificada).
Usuario: nombre del programador encargado de implementar la HU.
Prioridad del negocio: qué tan importante es para el cliente, se clasifica en Alta, Media y Baja.
Iteración asignada: iteración en la que se desarrollará la HU.
Puntos estimados: tiempo en semanas que se le asignará. (Estimado)
Descripción: es la descripción de la historia, detallando las operaciones del usuario y las respuestas del sistema.
Observaciones: informaciones de interés, como glosarios, detalles del usuario, entre otros.
Prototipo de Interfaz: contiene la imagen de una de las interfaces de usuario relacionadas con la HU.

Para dar cumplimiento al desarrollo de las funcionalidades del sistema fueron definidas 16 HU. A continuación se muestran las HU: Enviar inventario y Enviar trazas, por ser funcionalidades de prioridad alta para el cliente en el desarrollo de la aplicación. En el **¡Error! No se encuentra el origen de la referencia.** pueden encontrarse las HU restantes implementadas en el desarrollo de la aplicación.

Tabla 2: HU “Enviar inventario”

Historia de Usuario	
Número: HU-11	**Nombre de la Historia de Usuario:** Enviar inventario.

Modificación de Historia de Usuario Número: Ninguna	
Usuario: Rasiel Ernesto Ciervide Cabalé Yisel Elena Tamayo Betancourt	**Iteración Asignada:** 2
Prioridad en el negocio: Alta	**Puntos Estimados:** 1
Descripción: Consiste en enviar al servidor el inventario, en caso de un cambio en el inventario o en caso de que no exista inventario en el servidor.	
Observaciones:	
Prototipo de Interfaz: No procede.	

Tabla 3: HU "Enviar trazas"

Historia de Usuario	
Número: HU-13	**Nombre de la Historia de Usuario:** Enviar trazas.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Rasiel Ernesto Ciervide Cabale Yisel Elena Tamayo Betancourt	**Iteración Asignada:** 3
Prioridad en el negocio: Alta	**Puntos Estimados:** 0.4
Descripción: Consiste en enviar al servidor los datos de fecha y hora que se realizó el inventario cuando no se encuentra ninguna incidencia.	
Observaciones:	
Prototipo de Interfaz: No procede.	

2.5.2 Estimación del esfuerzo por HU

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto (máximo esfuerzo). Un punto, equivale a una semana ideal de programación donde se trabaje sin interrupciones. Las HU generalmente valen de 1 a 3 puntos, dependiendo de la complejidad de esta, teniendo en cuenta que el equipo de desarrollo valorará la duración de una HU teniendo en cuenta tres factores: la complejidad, el esfuerzo y el riesgo. Cuando una HU dura menos de una semana significa que está detallada al máximo y que puede ser combinada con otra. (11)

Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. (12)

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

La tabla 5 contiene la estimación del esfuerzo por HU según el orden a realizar.

Tabla 4: Estimación del esfuerzo por Historias de Usuarios.

Número	Historias de Usuarios	Puntos Estimados en días/semana
HU-1	Realizar inventario del hardware.	0.4
HU-2	Realizar inventario del CPU.	0.5
HU-3	Realizar inventario del GPU.	0.5
HU-4	Realizar inventario de la RAM.	0.2
HU-5	Realizar inventario del almacenamiento.	0.6
HU-6	Realizar inventario de la pantalla.	0.4
HU-7	Realizar inventario de las cámaras.	0.5
HU-8	Realizar inventario de la batería.	0.4
HU-9	Realizar inventario de los sensores.	0.5
HU-10	Realizar inventario del software y de las aplicaciones instaladas.	1
HU-11	Enviar inventario.	1
HU-12	Comprobar inventario y notificar incidencias.	0.6
HU-13	Enviar trazas.	0.4
HU-14	Configurar dirección del servidor.	1
Total		**40 días/8 semanas**

2.5.3 Plan de iteraciones

Esta etapa encierra varias iteraciones sobre el sistema antes de ser entregado. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: HU no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la

iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por la pareja de programadores. Teniendo en cuenta el riesgo para desarrollar cada una de las historias de usuario, el tamaño del equipo de desarrollo, así como otros factores subjetivos, se decidió dividir el proyecto en tres iteraciones, detalladas a continuación:

Iteración 1: En esta iteración se implementarán las HU de prioridad media, funcionalidades que permitirán obtener la información indispensable de la que no puede prescindir el inventario del dispositivo.

Iteración 2: En esta iteración se implementarán las HU de prioridad alta, las cuales constituyen funcionalidades indispensables para cubrir las necesidades del cliente.

Iteración 3: En esta iteración se implementarán las HU de prioridad baja, las cuales no son menos importantes que las mencionadas anteriormente pero debido al número de HU se determinó realizarlas en otra iteración.

La tabla 6 refleja la relación de HU que se realizarán por cada iteración, así como la duración total en semanas de estas. El tiempo estimado para cada iteración no supera el máximo establecido por la metodología XP (25) y la complejidad de las tareas permite agruparlas en el mismo orden del flujo de datos.

Tabla 5: Plan de Iteraciones.

Iteraciones	Orden de la HU por iteración	Duración Total de la Iteración en semanas
Iteración 1	HU-1, HU-4, HU-5, HU-6, HU-8, HU-10.	**3 semanas**
Iteración 2	HU-11, HU-12, HU-13, HU-14.	**3 semanas**
Iteración 3	HU-2, HU-3, HU-7, HU-9.	**2 semanas**
Total		**8 semanas**

2.5.4 Plan de entregas

El plan de entrega detalla la fecha límite o fecha de culminación de cada una de las iteraciones a realizar, una vez determinado el tiempo que demorará realizar cada una de las iteraciones será definido el próximo plazo de entrega.

En la siguiente tabla se muestra el Plan de Entregas por cada una de las iteraciones con las fechas aproximadas de entrega correspondientes.

Tabla 6: Plan de Entregas.

Herramienta	Final 1ra Iteración	Final 2da Iteración	Final 3ra Iteración
GClient para Android	27 de marzo de 2015	17 de abril de 2015	4 de mayo de 2015

Conclusiones del capítulo

Al concluir este capítulo fueron descritas las fases de Exploración y Planeación para la propuesta de solución, elemento que posibilitó la obtención de los artefactos necesarios para dar cumplimiento a la planificación establecida para el desarrollo del cliente del GRHS para dispositivos que operan con Android, en correspondencia a la metodología seleccionada tales como: historias de usuario, el esfuerzo estimado por HU, el plan de iteraciones así como el plan de entregas correspondiente para las iteraciones planteadas favoreciendo el orden en que se les dará cumplimiento según las necesidades del cliente.

El desarrollo de estas etapas permitió además que el proceso fuera dividido en tres iteraciones en las que se desarrollaron las HU de prioridad media durante la primera iteración por ser las que permiten obtener información indispensable para el control de los recursos de hardware y software en dispositivos que operan con Android, posteriormente fueron desarrolladas las de prioridad alta, estas HU son indispensables para el desarrollo del sistema y finalmente las de prioridad baja que brindan la información restante de las características de hardware del dispositivo. Determinándose que el proceso de desarrollo durará un total de 8 semanas, divididas en iteraciones de 3, 3 y 2 semanas respectivamente, de manera tal que el equipo de desarrollo y el cliente obtendrían un producto funcional al concluir cada iteración, posibilitando a este último plantear su conformidad o no con la aplicación obtenida.

Capítulo 3: “Iteraciones y producción.”

Introducción

En el presente capítulo se detallan las iteraciones desarrolladas durante la etapa de construcción de la propuesta de solución. Con este fin son definidas las tarjetas CRC como artefacto muy útil para respaldar el enfoque orientado a objetos. Son expuestas además las tareas de ingeniería resultantes por cada HU. Posteriormente se ejecutarán las pruebas que favorecen la buena calidad de la solución final. Los artefactos que se obtendrán serán el resultado del seguimiento de la filosofía determinada por la metodología XP, la cual propone que la implementación debe ejecutarse de manera iterativa e incremental, logrando al término de cada iteración un producto funcional que debe ser testeado de conjunto con el cliente; y a su vez es garantizada la retroalimentación entre los desarrolladores y el mismo. Además se define la arquitectura de la aplicación, estilos arquitectónicos, patrones de diseño y el modelo de datos y el diagrama de despliegue de la solución propuesta.

3.1 Arquitectura de software

La arquitectura de un software es el diseño de más alto nivel de la estructura de un sistema. Toda arquitectura de software debe definir diversos aspectos del software, y generalmente, cada uno de estos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. (26) La arquitectura determinada para el desarrollo del sistema GRHS es la arquitectura Cliente - Servidor.

La arquitectura Cliente - Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. (27) Se puede definir como una arquitectura distribuida en la que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes, permitiendo a los usuarios finales obtener acceso a la información de forma transparente. Ofrece ventajas de tipo organizativo debido a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La figura 8 describe la arquitectura Cliente - Servidor para la propuesta de solución.

Figura 8: Arquitectura Cliente – Servidor diseñada para la propuesta de solución.

Tal y como se muestra en la figura 8 el cliente va a ser el proceso que reclama los servicios de otro y permite al usuario formular los requerimientos y pasarlos al servidor. Normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI, por sus siglas en inglés), además de acceder a los servicios distribuidos en cualquier parte de una red. (28) Las funciones que lleva a cabo el proceso cliente para la presente investigación se resumen en los siguientes puntos:

- Recolectar la información de hardware del dispositivo.
- Recolectar la información de software del dispositivo.
- Construir el inventario de los recursos de hardware y software del dispositivo
- Guardar inventario en la base de datos local.
- Enviar inventario al servidor GRHS.
- Comprobar inventario.
- Enviar trazas.
- Notificar incidencias.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Recibir configuraciones del servidor.

Por otro lado, el servidor es el proceso que proporciona un servicio a otros, es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. (28)

Una de las ventajas que proporciona esta arquitectura es que al estar distribuidas las funciones y responsabilidades entre varios clientes independientes, es totalmente posible realizar cambios dígase reemplazar, reparar, actualizar, o incluso trasladar un servidor GRHS, mientras que sus clientes GRHS no se verán afectados por ese cambio, o en caso tal de que ocurra las afectaciones serían mínimas.

3.2 Estilo arquitectónico. Patrón de arquitectura.

Involucrados en una arquitectura, se encuentran los estilos arquitectónicos. Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software. (29)

Existen numerosos estilos arquitectónicos entre los que se encuentra el Estilo de llamada y retorno y dentro de él, el patrón de arquitectura Modelo-Vista-Controlador (MVC). Éste separa la lógica de control, la interfaz de usuario y los datos de una aplicación en tres componentes distintos. Se utiliza puesto que se tiene una capa de acceso a datos donde van a estar todas las clases que de una forma u otra piden información a una base de datos.

La capa Modelo compuesta por el paquete **cu.tlm.grhsclient.model** contiene todas las clases que tienen el código relacionado con el acceso a datos, para que este sea lo más genérico posible y se pueda reutilizar en otras situaciones y proyectos. Se incluirá n consultas a las bases de datos y validaciones de entrada de datos. Se evidencia dentro del modelo la clase DatabaseManager, la cual contiene varios métodos que envían a ejecutar consultas SQL al gestor obteniendo y devolviendo los resultados de las mismas para luego ser procesados.

La capa Vista compuesta por el paquete **cu.tlm.grhsclient.interfaces** contiene todas las clases que poseen el código representando la parte que será visualizada en pantalla por el usuario. Se evidencia dentro de este paquete las clases HomeInterface y GRHSClient, interfaces visuales que son mostradas al usuario para que el mismo interactúe con la aplicación.

La capa Controladora compuesta por el paquete **cu.tlm.grhsclient.utilities** que contiene las clases que ejecutan la lógica de la aplicación, realizan llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. Se evidencia en este paquete varias clases; AndroidInformation, AppInstalledAdapter, CheckDbChanges, ClientService, Conexion, ConexionHandler, Data, DataAdapter, JSONHandler, SimpleAdapter, SimpleData, WiFiMonitor, las cuales se encargan de recopilar toda la información y enviar los datos a las clases de la capa Vista y monitorear la conexión cliente -servidor.

Con el uso de este patrón se persigue mejorar la reusabilidad del código y que las modificaciones en las vistas impacten en menor medida en la lógica de negocio o de datos.

3.3 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Son a menudo mal interpretados como aplicables sólo a la programación en los grandes sistemas, pero en realidad, se pueden aplicar a la solución de problemas en la programación pequeña como en la implementación de estructuras de datos o algoritmos simples. (30) Los patrones de diseño se pueden combinar en los componentes que resuelven grandes problemas. Cabe destacar que existen varios grupos de patrones de diseño, pero la solución se basa en los más usados en función del objetivo que se ha planteado en este trabajo.

3.3.1 Patrones GRASP

Son denominados GRASP por las siglas en inglés de patrones generales de asignación de responsabilidades. Se basan en la determinación de las clases adecuadas y decidir cómo estas clases deben interactuar. Incluso cuando se utilizan metodologías rápidas como XP y el proceso se centra en el desarrollo continuo, es necesario elegir cuidadosamente las responsabilidades de cada clase desde la primera codificación y, fundamentalmente, en la refactorización del programa. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (31) A continuación se presentan los patrones GRASP empleados y su concreción en el diseño.

Patrón Experto

El experto en información es el encargado de definir las interacciones básicas entre los objetos, asignando responsabilidades a las clases de forma tal que el sistema sea más fácil de entender, mantener y ampliar, permitiendo reutilizar los componentes creados en futuras aplicaciones. (31) Es necesario asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. (32) La figura 9 evidencia el uso de este patrón en el desarrollo de la aplicación.

```
public class GPUFragment extends Fragment {

    ListView gpuList;
    private static SharedPreferences preferences;

    @Override
    public View onCreateView(LayoutInflater inflater,
            @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        // Auto-generated method stub
        View view = inflater.inflate(R.layout.gpu_layout, container, false);

        preferences = getActivity().getSharedPreferences("GPUinfo", Context.MODE_PRIVATE);
        gpuList = (ListView) view.findViewById(R.id.LstGPU);
        ArrayList<Data> gpuData = new ArrayList<Data>();
        DataAdapter gpu = new DataAdapter(getActivity(), gpuData);
        //String[] extensions = preferencias.getString("EXTENSIONS", null).split(" ");

        try {
            gpuData.add(new Data("Vendor", preferences.getString("VENDOR", null)));
            gpuData.add(new Data("Renderer", preferences.getString("RENDERER", null)));
            gpuData.add(new Data("Versi\u00f3n", preferences.getString("VERSION", null)));
            gpuData.add(new Data("Extensiones", preferences.getString("EXTENSIONS", null)));
        } catch (Exception e) {
            e.printStackTrace();
        }

        gpuList.setAdapter(gpu);

        return view;
    }
```

Figura 9: Implementación del patrón experto.

La clase GPUFragment es la encargada del comportamiento de la información del GPU del dispositivo. Posee la información vasta para la implementación del método onCreateView que tiene la responsabilidad de llenar los datos que van a ser mostrados en la vista y crearla.

Patrón Creador

El propósito fundamental de este patrón es asignar responsabilidades relacionadas con la creación de objetos producidos en cualquier evento. Indica que una clase es idónea para asumir la responsabilidad de crear el elemento registrado o contenido. (32) Una clase es responsable de crear una instancia de otra clase si: agrega objetos, contiene referencias a objetos, almacena instancias, utiliza estrechamente los objetos o tiene la información de inicialización que se necesita para crear un objeto de esa clase. Facilita un Bajo Acoplamiento, supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. La figura 10 muestra como fue empleado el patrón creador en la implementación de la aplicación.

```java
@DatabaseTable(tableName="apps")
public class Apps {

    private int id;

    private ForeignCollection<AppItem> items;

    public Apps() {}

    //Getter & Setter
    public int getId() {

    public void setId(int id) {

    public void setItems(ForeignCollection<AppItem> items) {

    public List<AppItem> getItems() {
        ArrayList<AppItem> itemList = new ArrayList<AppItem>();
        for (AppItem item : items) {
            itemList.add(item);
        }
        return itemList;
    }
}
```

Figura 10: Implementación del patrón creador.

La clase Apps es la encargada de la creación de objetos de la clase AppItem en una agregación de uno a muchos.

Patrón Alta cohesión

Asigna a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad, evitando así que una clase sea la única responsable de muchas tareas en áreas funcionales muy heterogéneas. La información que maneja una entidad de software tiene que estar conectada lógicamente con esta, no deben existir entidades con atributos que describan comportamientos que en realidad no le corresponden. (31) Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo.

La clase Monitor es la clase modelo que permite obtener las características de la pantalla del dispositivo y modificarlas, colabora con las clases DatabaseHelper brindándole la información necesaria de los datos de la pantalla. La figura 10 muestra el uso de este patrón.

```
public class Monitor {

    private int id;

    private int horizontalSize;

    private int verticalSize;

    private String dpiDensity;

    public Monitor() {}

    //Getter & Setter
    public int getId() {

    public void setId(int id) {

    public int getHorizontalSize() {

    public void setHorizontalSize(int horizontalSize) {

    public int getVerticalSize() {

    public void setVerticalSize(int verticalSize) {

    public String getDpiDensity() {
        return dpiDensity;
    }

    public void setDpiDensity(String dpiDensity) {
        this.dpiDensity = dpiDensity;
    }
}
```

Figura 11: Implementación del patrón alta cohesión.

Patrón Bajo acoplamiento

Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente como Experto o Alta Cohesión. El acoplamiento tal vez no sea tan importante, sino se busca la reutilización. No existe una medida absoluta de cuando el acoplamiento es excesivo. En términos generales, han de tener escaso acoplamiento las clases muy genéricas y con grandes probabilidades de reutilización. (31)

El diseño propuesto respeta este patrón al evidenciar clases con el menor número de clases relacionadas posible. Ejemplo de este patrón son las clases WiFiMonitor y ConexionHandler.

Patrón Controlador

El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. (31)

La clase AndroidInformation muestra el empleo de este patrón en el diseño de la propuesta de solución. Tiene la responsabilidad de recolectar toda la información de hardware y software del dispositivo implementando las funcionalidades que permiten obtener del resto de las clases las características de cada uno de los componentes. Su función controladora le permite delegar en otros objetos el trabajo que se necesita hacer, sólo coordina y controla la actividad de recolección de información.

3.3.2 Patrones de Concurrencia

Los patrones de esta categoría permiten coordinar las operaciones concurrentes. Estos patrones se dirigen principalmente a dos tipos diferentes de problemas (33):

1. Recursos compartidos: Cuando las operaciones concurrentes acceden a los mismos datos u otros tipos de recursos compartidos, podría darse la posibilidad de que las operaciones interfirieran unas con otras si ellas acceden a los recursos al mismo tiempo. Para garantizar que cada operación se ejecuta correctamente, la operación debe ser protegida para acceder a los recursos compartidos en solitario. Sin embargo, si las operaciones están completamente protegidas, entonces podrían bloquearse y no ser capaces de finalizar su ejecución. El bloqueo es una situación en la cual una operación espera por otra para realizar algo antes de que esta proceda. Porque cada operación está esperando por la otra para hacer algo, entonces ambas esperan para siempre y nunca hacen nada.

2. Secuencia de operaciones: Si las operaciones son protegidas para acceder a un recurso compartido una cada vez, entonces podría ser necesario garantizar que ellas acceden a los recursos compartidos en un orden particular. Por ejemplo, un objeto nunca será borrado de una estructura de datos antes de que esté sea añadido a la estructura de datos.

Este patrón es usado en la clase HomeInterface, de forma tal que una vez levantada la interfaz principal del sistema se arranca un hilo concurrente al principal en el cual se obtiene la MAC de la Wi-Fi y la información de las cámaras, esto se hace debido a que para recoger esta información del sistema el dispositivo tiene que iniciar el sensor de la cámara y levantar la Wi-Fi, lo que generaba una demora a la hora de abrir las pestañas en la interfaz que contenía dicha información, habiendo hecho este proceso de recopilación de datos en un hilo aparte la velocidad de la aplicación aumentó considerablemente.
También se usa en la clase ClientService para realizar las operaciones relacionadas con la transmisión de datos entre el cliente y el servidor, debido a que a partir de la versión 4.0 de Android las operaciones relacionadas con la red no pueden realizarse en el hilo principal, deben hacerse en un hilo secundario.

3.4 Modelo de Datos

El modelado de datos es el lenguaje empleado para realizar la representación de una base de datos. Se define modelo de datos como el conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los modelos de datos comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones.

El modelo entidad-atributo-valor es la representación de un modelo flexible donde se pueden representar objetos con sus atributos, es un acercamiento al modelo orientado a objetos representado en el modelo relacional, donde la entidad Component representa las clases o los componentes que integran el dispositivo, la entidad Attribute representa los atributos de los componentes, mientras que la entidad Value representa los valores de cada atributo para cada objeto dado. En la figura 12 se representa el modelo de datos para la propuesta de solución.

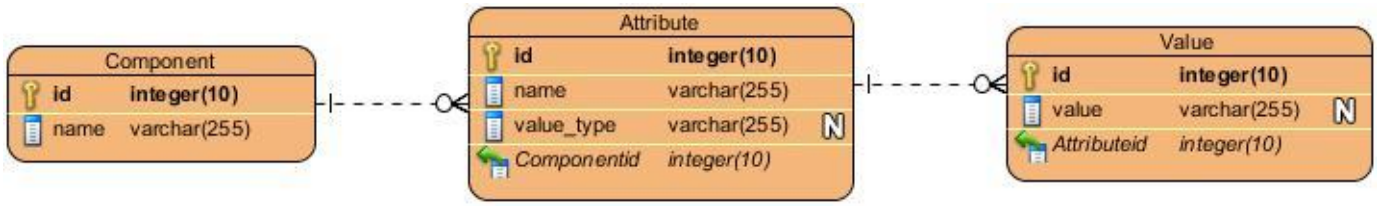


Figura 12: Modelo de datos de la propuesta de solución.

3.5 Diagrama de Despliegue

El diagrama de despliegue es utilizado para realizar la distribución física de un sistema. El mismo muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los link de la comunicación entre ellos y las instancias de los componentes y objetos que residen en ellos (34). Para el cliente GRHS para dispositivos que operan con Android el diagrama de despliegue es el que se muestra en la figura 10.

Descripción de los nodos del diagrama de despliegue:

Estación Cliente: Dispositivo con sistema operativo Android instalado con versión 2.2 o superior. En este nodo es donde estará la aplicación, su función es la ejecución del sistema e interactuar con el mismo según las necesidades del usuario. En ella estará la base de datos de la aplicación en un fichero.

Estación Servidor: Servidor GRHS es el nodo donde se encuentra el servidor y donde se encuentra todo lo necesario para recibir el inventario recolectado por el GClient para dispositivos que operan con Android, en ella se encuentra la parte web del sistema.

Estación Servidor Database: Es el nodo donde se encuentra la base de datos Postgres del sistema GRHS.

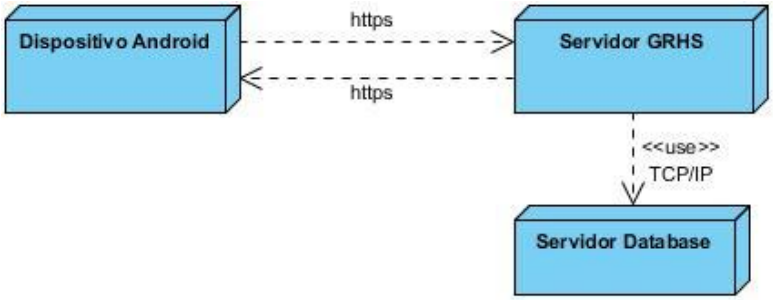


Figura 13: Diagrama de despliegue para la propuesta de solución.

3.6 Tarjetas CRC (Clase, Responsabilidad y Colaboración)

El desarrollo de cualquier proyecto requiere de un buen diseño de sus clases para de esta forma realizarlo con la mejor calidad posible y así el cliente quede satisfecho. En la metodología XP el diseño de las clases se realiza a través de las tarjetas CRC, para de esta forma ayudar al refinamiento de las clases. Sirven para diseñar el sistema en conjunto entre todo el equipo de desarrollo, aunque su principal objetivo es propiciar el enfoque orientado a objetos y reducir el modo de pensar procedimental. Están diseñadas en cuatro secciones: nombre de la clase, descripción, responsabilidades y colaboradores.

Una clase describe un objeto o evento del sistema, mediante sus atributos y métodos. Las responsabilidades de estas se describen por las tareas que realiza o por los métodos y los colaboradores son las demás clases con las que interactúa para cumplir con sus responsabilidades. A continuación se describen las tarjetas CRC diseñadas para la implementación del cliente GRHS para dispositivos que operan con Android.

Tabla 7: Tarjeta CRC "Clase: DatabaseHelper."

Clase: DatabaseHelper	
Descripción: Encargada de interactuar directamente con la base de datos.	
Responsabilidad	**Colaboradores**

Crear las tablas. Implementar los métodos necesarios para que la clase DatabaseManager realice operaciones sobre la base de datos.	App, Battery, BuildInfo, Camera, GPU, GPUExtensionItem, Line, Microprocessor, Monitor NetworkInterface, RAM, Sensors Software, StorageDevice, Database Manager

Tabla 8: Tarjeta CRC "Clase: DatabaseManager"

Clase: DatabaseManager	
Descripción: Realizar consultas sobre la base de datos	
Responsabilidad	**Colaboradores**
Es la clase encargada de agregar, eliminar, modificar y realizar las operaciones sobre la base de datos	App, Battery, BuildInfo, Camera, GPU, GPUExtensionItem, Line, Microprocessor, Monitor NetworkInterface, RAM, Sensors Software, StorageDevice, CheckDbChanges, ClientService

Tabla 9: Tarjeta CRC "Clase: StorageFragment."

Clase: StorageFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información de almacenamiento del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información del almacenamiento del dispositivo brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

Tabla 10: Tarjeta CRC "Clase: AppInstalledFragment."

Clase: AppInstalledFragment	
Descripción: Se encarga de crear la interfaz para mostrar las aplicaciones instaladas en el dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información de las aplicaciones instaladas en el dispositivo. Pasar por parámetro esta información a la clase AppInstalledAdapter para construir la interfaz.	AppInstalledAdapter

Tabla 11: Tarjeta CRC "Clase: BatteryFragment."

Clase: BatteryFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información de la batería del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información de la batería brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

Tabla 12: Tarjeta CRC "Clase: CameraFragment."

Clase: CameraFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información de la(s) cámara(s) del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información de la(s) cámaras(s) brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

Tabla 13: Tarjeta CRC "Clase: CPUFragment."

Clase: CPUFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información del CPU del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información del CPU brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

Tabla 14: Tarjeta CRC "Clase: GPUFragment."

Clase: GPUFragment
Descripción: Se encarga de crear la interfaz para mostrar la información del GPU del dispositivo.

Responsabilidad	Colaboradores
Obtener la información del GPU del dispositivo. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	DataAdapter

Tabla 15: Tarjeta CRC "Clase: GRHSClient."

Clase: GRHSClient	
Descripción: Se encarga de crear la interfaz principal de la aplicación.	
Responsabilidad	**Colaboradores**
Crea la interfaz principal de la aplicación. Permitir el desplazamiento por las distintas pestañas de las interfaces.	AdaptadorDeFragmento

Tabla 16: Tarjeta CRC "Clase: BuildFragment."

Clase: BuildFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información del hardware del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información del hardware brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

Tabla 17: Tarjeta CRC "Clase: DisplayFragment."

Clase: DisplayFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información de la pantalla del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información de la pantalla brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

Tabla 18: Tarjeta CRC "Clase: RamFragment."

Clase: RamFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información de la memoria RAM del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información de la memoria RAM brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

Tabla 19: Tarjeta CRC "Clase: SensorsFragment."

Clase: SensorsFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información de los sensores del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información de los sensores brindada por AndroidInformation. Pasar por parámetro esta información a la clase SimpleAdapter para construir la interfaz.	AndroidInformation SimpleAdapter

Tabla 20: Tarjeta CRC "Clase: SoftwareFragment."

Clase: SoftwareFragment	
Descripción: Se encarga de crear la interfaz para mostrar la información del software del dispositivo.	
Responsabilidad	**Colaboradores**
Obtener la información del software brindada por AndroidInformation. Pasar por parámetro esta información a la clase DataAdapter para construir la interfaz.	AndroidInformation DataAdapter

3.7 Tareas de ingeniería

Las tareas de ingeniería se encuentran asociadas fuertemente a las funcionalidades y características que se deben cumplir. Pertenecen a una HU en específico. Se consideran como la entrada de trabajo para el equipo de programadores. Es una ficha que contiene el número identificador de la tarea, el identificador de

la HU con la que está relacionada, el nombre de la tarea, la fecha de inicio, la fecha de fin, el programador responsable y la descripción.

3.7.1 Iteración 1

Durante la primera iteración se abordaron las HU de prioridad media donde se construyó la base arquitectónica del producto con las funcionalidades fundamentales y necesarias para ser mostradas al cliente y obtener una retroalimentación rápida y amplia. En la tabla 21 se describe las HU correspondientes a esta iteración el tiempo estimado y el tiempo real utilizado para la realización de las tareas de ingeniería.

Tabla 21: HU implementadas durante la primera iteración

Historia de Usuario	Estimación en días	Real en días
Realizar inventario de hardware.	2	2
Realizar inventario de la RAM.	1	1
Realizar inventario del almacenamiento.	3	3
Realizar inventario de la pantalla.	2	2
Realizar inventario de la batería.	2	2
Realizar inventario del software y las aplicaciones instaladas.	5	5

En la tabla 22 se muestran las tareas de ingeniería a implementar por cada HU correspondiente a esta iteración.

Tabla 22: Tareas de ingeniería a desarrollar durante la primera iteración.

Historias de Usuario	Tareas
Realizar inventario de hardware.	Implementar realizar inventario de hardware.
Realizar inventario de la RAM.	Implementar realizar inventario de la RAM.
Realizar inventario del almacenamiento.	Implementar realizar inventario de almacenamiento interno. Implementar realizar inventario de almacenamiento externo.
Realizar inventario de la pantalla.	Implementar realizar inventario de la pantalla.

Realizar inventario de la batería.	Implementar realizar inventario de la batería.
Realizar inventario del software y las aplicaciones instaladas.	Implementar realizar inventario del software. Implementar realizar inventario de las aplicaciones.

A continuación en la tabla 23 se refleja una muestra de las tareas de la ingeniería asociadas a la HU "Realizar inventario de almacenamiento" perteneciente a la primera iteración. El resto de las tareas pueden ser localizadas en el **Anexo II: Tareas de Ingeniería** del presente trabajo.

Tabla 23: Tarea # 3 "Implementar realizar inventario de almacenamiento interno"

Tarea de Ingeniería	
Número de Tarea:3	**Nombre de la Historia de Usuario:** Realizar inventario del almacenamiento.
Nombre de la Tarea: Implementar realizar inventario de almacenamiento interno.	
Tipo de Tarea: Desarrollo	**Puntos Estimados: 0.4/1**
Fecha de Inicio: 12-03-2015	**Fecha Fin:** 13-03-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan obtener las características del almacenamiento interno (capacidad de almacenamiento del sistema, capacidad libre de almacenamiento del sistema, capacidad de almacenamiento interno emulado si tiene, capacidad libre de almacenamiento interno emulado si tiene).	

Tabla 24: Tarea # 4 "Implementar realizar inventario de almacenamiento interno"

Tarea de Ingeniería	
Número de Tarea:4	**Nombre de la Historia de Usuario:** Realizar inventario del almacenamiento.
Nombre de la Tarea: Implementar realizar inventario de almacenamiento externo.	
Tipo de Tarea: Desarrollo	**Puntos Estimados: 0.2/1**
Fecha de Inicio: 16-03-2015	**Fecha Fin:** 16-03-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan obtener las características del almacenamiento externo (si tiene tarjeta micro SD y en caso de tener obtener la capacidad de almacenamiento de la tarjeta micro SD, capacidad libre de almacenamiento de la tarjeta micro SD) del	

dispositivo.

3.7.2 Iteración 2

Durante esta iteración se abordaron las HU de alta prioridad. Al culminar se consta de un producto casi listo para su puesta en funcionamiento con la mayoría de sus funcionalidades más críticas ya implementadas. A continuación se describe en la tabla 25 las HU implementadas con la estimación y el tiempo real de cada una:

Tabla 25: HU implementadas durante la segunda iteración.

Historia de Usuario	Estimación en días	Real en días
Enviar inventario.	5	5
Comprobar inventario y notificar incidencias.	3	3
Enviar trazas.	2	2
Configurar dirección del servidor.	5	5

Las tareas de ingeniería a desarrollar por cada una de las HU anteriores son:

Tabla 26: Tareas de ingeniería para la segunda iteración.

Historias de Usuario	Tareas
Enviar inventario	Implementar monitorear conexión. Implementar ejecutar servicio. Implementar obtener inventario. Implementar enviar inventario.
Comprobar inventario y notificar incidencias	Implementar comprobar inventario. Implementar notificar incidencias.
Enviar trazas	Implementar enviar trazas.
Configurar dirección del servidor	Implementar obtener datos insertados por el usuario. Implementar comprobación de los datos. Implementar guardar datos en las preferencias compartidas.

A continuación se detallan las tareas de ingeniería relacionadas con la HU “Enviar inventario”, para la iteración 2, el resto de las tareas de esta iteración se muestran en el ¡Error! No se encuentra el origen de la referencia.

Tabla 27: Tarea # 9 "Implementar monitorear conexión"

Tarea de Ingeniería	
Número de Tarea:9	**Nombre de la Historia de Usuario:** Enviar inventario.
Nombre de la Tarea: Implementar monitorear conexión.	
Tipo de Tarea: Desarrollo	**Puntos Estimados: 0.4/1**
Fecha de Inicio: 30-03-2015	**Fecha Fin:** 31-03-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan comprobar que el dispositivo se encuentra conectado a la red y en caso de estar conectado permite el Cliente GRHS en el dispositivo.	

Tabla 28: Tarea # 10 "Implementar ejecutar servicio"

Tarea de Ingeniería	
Número de Tarea:10	**Nombre de la Historia de Usuario:** Enviar inventario.
Nombre de la Tarea: Implementar ejecutar servicio.	
Tipo de Tarea: Desarrollo	**Puntos Estimados: 0.2/1**
Fecha de Inicio: 01-04-2015	**Fecha Fin:** 01-04-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan comprobar que una vez conectado el dispositivo a la red se inicie el servicio, y se publique una notificación informando que el servicio se encuentra en ejecución.	

Tabla 29: Tarea # 11 "Implementar obtener inventario"

Tarea de Ingeniería	
Número de Tarea:11	**Nombre de la Historia de Usuario:** Enviar inventario.
Nombre de la Tarea: Implementar obtener inventario.	
Tipo de Tarea: Desarrollo	**Puntos Estimados: 0.2/1**
Fecha de Inicio: 02-04-2015	**Fecha Fin:** 02-04-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan obtener los inventarios de hardware y	

software (toda la información que ofrece el dispositivo), transformar y guardar toda la información obtenida en un formato JSON para poder transferirlo al servidor.

Tabla 30: Tarea # 12 "Implementar enviar inventario"

Tarea de Ingeniería	
Número de Tarea:12	**Nombre de la Historia de Usuario:** Enviar inventario.
Nombre de la Tarea: Implementar enviar inventario	
Tipo de Tarea: Desarrollo	**Puntos Estimados: 0.2/1**
Fecha de Inicio: 03-04-2015	**Fecha Fin:** 03-04-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan guardar el inventario obtenido en la BD, comprobar el IP de envío del inventario y enviar el inventario al servidor.	

3.7.3 Iteración 3

En esta iteración serán implementadas las HU de la tercera iteración cuyo resultado final será un producto listo para su funcionamiento.

Tabla 31: HU implementadas durante la tercera iteración.

Historia de Usuario	Estimación en días	Real en días
Realizar inventario del GPU.	2.5	2.5
Realizar inventario del CPU.	2.5	2.5
Realizar inventario de las cámaras.	2.5	2.5
Realizar inventario de los sensores.	2.5	2.5

Las tareas a desarrollar para las HU anteriores son:

Tabla 32: Tareas de ingeniería para la tercera iteración.

Historias de Usuario	Tareas
Realizar inventario del GPU	Implementar realizar inventario del GPU
Realizar inventario del CPU.	Implementar realizar inventario del CPU.
Realizar inventario de las cámaras.	Implementar realizar inventario de la cámara trasera.

	Implementar realizar inventario de la cámara frontal.
Realizar inventario de los sensores.	Implementar realizar inventario de los sensores.

A continuación se detallan algunas de las tareas relacionadas con la iteración 3, el resto aparecen en el ¡Error! No se encuentra el origen de la referencia.

Tabla 33: Tarea # 19 "Implementar realizar inventario del GPU."

Tarea de Ingeniería	
Número de Tarea: 19	**Nombre de la Historia de Usuario:** Realizar inventario del GPU.
Nombre de la Tarea: Implementar realizar inventario del GPU.	
Tipo de Tarea: Desarrollo	**Puntos Estimados: 0.2/1**
Fecha de Inicio: 20-04-2015	**Fecha Fin:**20-04-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan obtener las características del GPU del dispositivo (renderer, versión y proveedor).	

Tabla 34: Tarea # 20 "Implementar realizar inventario del CPU."

Tarea de Ingeniería	
Número de Tarea:20	**Nombre de la Historia de Usuario:** Realizar inventario del CPU.
Nombre de la Tarea: Implementar realizar inventario del CPU.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:0.4/1**
Fecha de Inicio: 21-04-2015	**Fecha Fin:** 22-04-2015
Programador(es) Responsable(s): Rasiel Ernesto Ciervide Cabalé, Yisel Elena Tamayo Betancourt	
Descripción: Se implementarán las funcionalidades que permitan obtener las características del CPU del dispositivo (arquitectura, procesador, revisión del CPU, hardware del CPU, frecuencia máxima y actual del CPU).	

3.8 Pruebas al sistema

La realización de pruebas permite comprobar la eficacia de un sistema, estas son responsables de la verificación del cumplimiento de los objetivos trazados en la etapa de implementación. Con las pruebas se

reduce el número de errores durante la implementación, el tiempo entre la introducción de estos en el sistema y su detección; son las encargadas de aumentar la seguridad y de evitar efectos colaterales no deseados a la hora de realizar modificaciones en la aplicación.

Conforme a la metodología XP, se lleva a cabo la Fase de Prueba que establece probar constantemente como sea posible, permitiendo un aumento de la calidad del sistema desarrollado y reduciendo el número de errores no detectados. XP es una metodología que divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación.

La estrategia de prueba define las técnicas de prueba que se van a emplear (manual o automática) así como las herramientas que serán empleadas en las mismas. Además de los criterios de éxitos y culminación de las pruebas que serán precisados, tanto como las consideraciones afectadas por los requerimientos de recursos o que tengan implicaciones en la planificación. La estrategia a seguir para la realización de las pruebas al cliente del GRHS para dispositivos con Android implementado contempla dos niveles de pruebas: pruebas de Unidad y pruebas de Aceptación.

3.8.1 Pruebas Unitarias

Las pruebas unitarias son la verificación de una unidad de código del software. Aplicables a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos funcionen como se espera. Las pruebas de unidad siempre están orientadas a caja blanca y le permiten al programador conocer si determinada funcionalidad se puede agregar al sistema sin afectar su funcionamiento.

XP propone realizar las HU en pequeñas iteraciones para permitir aumentar la calidad del sistema y reducir el número de errores no detectados disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. Por lo que se decidió dividir el proceso de desarrollo del cliente del GRHS para dispositivos que operan en Android en tres iteraciones, realizándose, al final de cada iteración, pruebas al código de las HU implementadas y verificar así su correcto funcionamiento.

Para la realización de estas pruebas se utilizó el framework JUnit 3.0 que está integrado con el SDK de Android. Este framework permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Por lo que todo el código debe tener pruebas unitarias y debe pasarlas antes de ser integrado al sistema GRHS.

Pruebas unitarias primera iteración

Al culminar la implementación de las HU de la primera iteración se le realizaron las pruebas al código arrojando como no conformidades:

- No se reconocía el modelo del dispositivo.
- No se reconocía la existencia de Micro SD.
- Los valores presentados por el almacenamiento interno no se correspondían con las propiedades reales de almacenamiento en el dispositivo.
- No se actualizaba la lista de aplicaciones instaladas.

A estas deficiencias se les dio solución, obteniendo finalmente resultados alentadores en iteraciones de prueba posteriores.

Pruebas unitarias segunda iteración

Se realizaron las pruebas al código de las HU de la segunda iteración detectándose las siguientes deficiencias:

- No se notificaban correctamente las incidencias detectadas.
- Existencia de problemas en la configuración de la dirección del servidor.

Después de haber solucionado las no conformidades se realizaron varias iteraciones de pruebas automáticas arrojando resultados positivos.

Pruebas unitarias tercera iteración

Fueron realizadas las pruebas unitarias a las HU de la tercera iteración detectándose las siguientes no conformidades:

- No se mostraba la lista de extensiones del GPU.
- No se mostraban los datos de la cámara frontal.

A estos problemas se les dio solución y en las siguientes etapas de esta iteración de pruebas se alcanzaron los criterios de éxito esperados. Para ver resultados de las pruebas realizadas consultar **Anexo IV: Pruebas unitarias.**

3.8.2 Pruebas de Aceptación

Las pruebas funcionales o de aceptación son especificadas por el cliente, se centran en los requerimientos funcionales del software y van dirigidas a las características generales y las funcionalidades del sistema. Son realizadas por el usuario final permitiendo la valoración del producto, donde el cliente confirma que las funcionalidades exigidas y descritas en la HU funcionan correctamente. Intentan encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos, errores de rendimiento y errores de inicialización y terminación. (24) Cuando se pasa la prueba de aceptación se considera la historia de usuario finalizada.

Las pruebas de aceptación se realizaron con el objetivo de verificar que el producto cumple con los requerimientos establecidos por el cliente, por lo que este constituyó una parte primordial en el desarrollo de las mismas. Las pruebas se llevaron a cabo con el cliente para dispositivos Android integrado al sistema GRHS, dándole un uso real a la propuesta de solución. Los errores detectados fueron reportados como no conformidades.

Para el desarrollo de las pruebas se emplearon los casos de prueba de aceptación donde se describe el proceso de prueba a realizar. Algunas de las deficiencias identificadas en las iteraciones fueron las siguientes:

- No se mostraban los datos de almacenamiento externo del dispositivo.
- No se mostraba el espacio disponible de almacenamiento interno.
- No se mostraban los datos de las cámaras.
- No se mostraba la lista de sensores del dispositivo.
- No se reconocía la densidad DPI de la pantalla.

Las no conformidades detectadas por el cliente fueron solucionadas en las iteraciones de pruebas. Las no conformidades detectadas por el cliente fueron solucionadas en las iteraciones de prueba. Además se probó la propuesta de solución en variadas ocasiones, formando el cliente una parte significativa de las pruebas y obteniéndose resultados satisfactorios. Los resultados finales de las iteraciones de pruebas de aceptación se muestran en el **Anexo III: Pruebas de Aceptación** y se resumen en la figura 11.

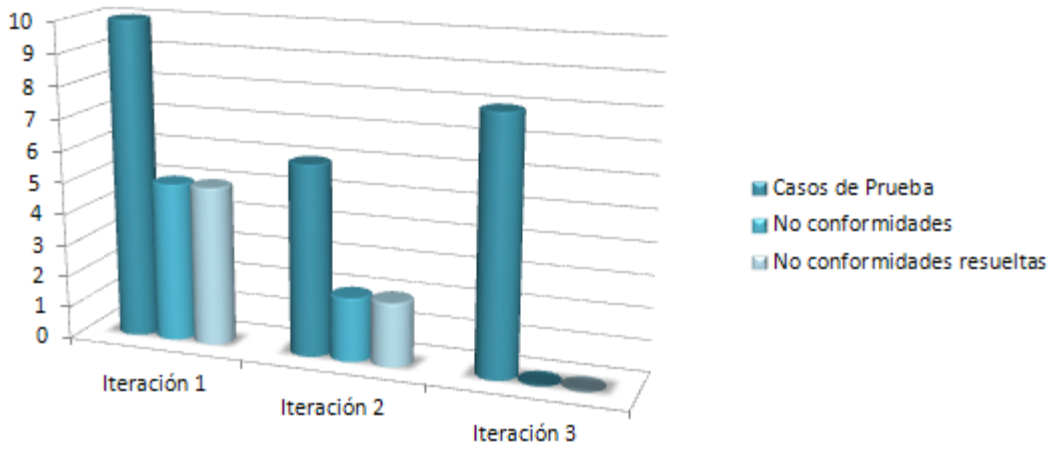


Figura 14: Resultados por iteraciones de las pruebas aplicadas.

Conclusiones del Capítulo

Una vez finalizado el capítulo donde se enmarcan las etapas de Iteraciones y Producción correspondiente a la metodología de desarrollo seleccionada se arribó a las siguientes conclusiones:

- La definición de la arquitectura, los patrones y estilos arquitectónicos permitió la creación de una aplicación cuyo código será legible y entendible para los desarrolladores en próximas versiones.
- La confección de las tarjetas CRC estableció el marco de relaciones para las clases del diseño, posibilitando una comprensión en la colaboración entre clases.
- La elaboración de las tareas de ingeniería facilitó la concreción de cada una de las actividades asociadas a las HU, permitiendo la implementación de estas.
- La ejecución de las pruebas de unidad al código de la aplicación definidas por los desarrolladores así como de las pruebas de aceptación conjuntamente con el cliente, permitieron presentar los resultados arrojados en cada iteración logrando finalmente una aplicación que responde al conjunto de funcionalidades identificadas.

Conclusiones Generales

El desarrollo del presente trabajo de diploma permitió que se llevaran a cabo todas las tareas de la investigación a fin de dar cumplimiento al objetivo general logrando arribar a las siguientes conclusiones:

- La elaboración del marco teórico de la investigación a través de la definición de conceptos fundamentales para su comprensión y el estudio del estado del arte realizado, de las soluciones informáticas existentes dedicadas a la gestión de información de recursos de hardware y software en dispositivos que operan con Android, permitió al equipo de desarrollo establecer un punto de partida de la posición actual del mundo respecto a la gestión de recursos de hardware y software para este tipo de medios, logrando identificar posibles funcionalidades para el sistema implementado. Denotando la no existencia de una herramienta en la actualidad capaz de integrarse al sistema GRHS que facilite la gestión de recursos de hardware y software para dispositivos Android.
- Las etapas de Exploración y Planificación definidas por la metodología de desarrollo seleccionada permitieron al equipo de desarrollo cumplir con el plan de entrega predefinido con el cliente, permitiendo el desarrollo de la aplicación por iteraciones y la integración de un producto final completamente funcional.
- El desarrollo del GClient para Android permitió al equipo de desarrollo la creación de una aplicación para gestionar los recursos de hardware y software a nivel institucional. Una vez integrada al sistema GRHS, la herramienta desarrollada permitió la elaboración de inventarios de recursos de hardware y software sobre los dispositivos que operan bajo la plataforma Android y que sirven al centro actualmente.

Por todo lo expuesto, se concluye que el objetivo propuesto para el presente trabajo de diploma se ha cumplido satisfactoriamente, poniendo en práctica todas las tareas presentadas para el desarrollo del cliente del GRHS para dispositivos que operan con Android.

Recomendaciones

Con el objetivo de mejorar el funcionamiento y la utilidad del Cliente del GRHS para dispositivos que operan con Android se recomienda:

- Añadir las funcionalidades de detección del tipo de incidencias y funcionalidades de ejecución de acciones ante las incidencias detectadas en versiones superiores de la aplicación.
- Implantar la solución implementada en aquellas instituciones del país que empleen dispositivos que operan con Android como recurso institucional, con el objetivo de mejorar la gestión de inventario de los recursos disponibles.

Referencias Bibliográficas

1. **Alegsa, Leandro.** Diccionario de Informática y Tecnología. *Diccionario de Informática y Tecnología.* [En línea] [Citado el: 1 de Abril de 2015.] http://www.alegsa.com.ar/Dic/software.php.

2. **Brand Speaker, S.L.** TICbeat. [En línea] [Citado el: 24 de enero de 2015.] http://www.ticbeat.com/tecnologias/android-domina-mercado-mundial-smartphones/.

3. **AB Internet Networks 2008 S.L.** AndroidSIS. [En línea] [Citado el: 2 de abril de 2015.] http://www.androidsis.com/la-verdadera-historia-de-android-nacimiento-del-sistema-operativo-2003/.

4. **Huice, Ing. Odaysa Rodriguez.** *Programa de Mejora 0208_Proyecto Técnico v1.0 Proyecto Gestión de Recursos de Hardware y Software.*

5. **Carlota Bustelo Ruesta, Raquel Amarilla Iglesias.** InterContact: Gestión del Conocimiento. [En línea] [Citado el: 1 de Abril de 2015.] comunidad/archivos/Gestion_del_Conocimiento-BusteloRuesta-AmarillaIglesias.pdf.

6. **Real Academia Española.** Real Academia Española. *Diccionario de la Lengua Española.* [En línea] [Citado el: 1 de Abril de 2015.] http://lema.rae.es/drae/?val=hardware.

7. **Digital Millennium Copyright Act (DMCA).** APPSAPK. *System Info Droid Android.* [En línea] [Citado el: 12 de Octubre de 2014.] http://www.appsapk.com/system-info-droid/.

8. **Google Inc.** Assistant for Android. *Google Play.* [En línea] [Citado el: 12 de Octubre de 2014.] https://play.google.com/store/apps/details?id=com.advancedprocessmanager&hl=es_419.

9. **FinalWire Ltd.** Aida64. [En línea] [Citado el: 1 de Mayo de 2015.] http://www.aida64.com.

10. **Google Inc.** AIDA64 Aplicaciones Android. *Google Play.* [En línea] FinalWire LTD., 27 de Abril de 2015. [Citado el: 1 de Mayo de 2015.] https://play.google.com/store/apps/details?id=com.finalwire.aida64&hl=es_419.

11. **Change., Extreme Programming Explained: Embrace.** *Kent Beck.* s.l. : Addison Wesley, 1999.

12. **Patricio Letelier, Mª Carmen Penadés.** *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Universidad Politécnica de Valencia : s.n.

13. **Gregory, S. Parallel.** *Logic Programming in PARLOG. The language and its implementation.* s.l. : Addison Wesley, 1987.

14. **Horstman Cay S., Gary Cornell.** *Core Java 2 Volume I Fundamentals.* 2000.

15. **Eclipse Foundation.** About the Eclipse Foundation. *Eclipse.* [En línea] [Citado el: 23 de febrero de 2015.] http://www.eclipse.org/org/.

16. **Chavarría, Raúl Eduardo.** IDE Eclipse Breve Guía. *Slideshare.* [En línea] 12 de Diciembre de 2007. [Citado el: 13 de Marzo de 2015.] http://www.slideshare.net/Benedeti/ide-eclipse-breve-gua-201399.

17. **Google Inc.** ADT Plugins Release Notes. *Android Developers.* [En línea] [Citado el: 28 de marzo de 2015.] http://developer.android.com/intl/es/tools/sdk/eclipse-adt.html.

18. **The JSON Data Interchange Standard.** Introducing JSON. [En línea] [Citado el: 13 de Marzo de 2015.] http://www.json.org/. ECMA-404 .

19. **Navathe, S. and R.Elmasri.** *Fundamentos de Sistemas de Bases de Datos.* 2002.

20. **Maldonado, Daniel Martin.** Empresa y Economía. SQLite, el motor de base de datos ágil y robusto. [En línea] 2008. [Citado el: 2015 de 13 de marzo.] http://www.empresayeconomia.es/aplicaciones-para-empresas/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.

21. **Flores Cueto, Juan José y Bertolotti Zuñiga, Carmen.** Diagrama de clases en UML. *Diagrama de clases en UML.* [En línea] [Citado el: 14 de Abril de 2015.] http://es.scribd.com/doc/31096724/Diagrama-de-Clases-en-UML#scribd.

22. **Visual Paradigm Copyright.** UML, BPMN and Software Design Tools for Agile Teams. [En línea] [Citado el: 2 de diciembre de 2014.] http://www.visual-paradigm.com/.

23. **Pressman, Roger S.** *Ingeniería del Software : Un Enfoque Práctico 6ta Edición.* 1998.

24. **Scott Hommel Sun Microsystems Inc.** *Convenciones de Código para el lenguaje de programación JAVA.* 2001.

25. **Galiano, Fernando Berzal.** *Diseño de Arquitecturas de Software.*

26. **Bertha Mariel Márquez Avendaño, José Manuel Zulaica Rugarcía.** Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español . *Colección de Tesis Digitales Universidad de las Américas Puebla.* [En línea] [Citado el: 30 de marzo de 2015.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/.

27. **Reynoso, Carlos Billi.** *Introducción a la arquitectura de software Versión 1.0.* Buenos Aires : Universidad de Buenos Aires, 2004.

28. **García Carmona, Juan.** GRASP: Alta cohesión y bajo acoplamiento . *Juan García Carmona. Metodologías ágiles, testing y desarrollo de software.* [En línea] 7 de Septiembre de 2012. [Citado el: 14 de Abril de 2015.] http://juan-garcia-carmona.blogspot.com/2012/09/grasp-alta-cohesion-y-bajo-acoplamiento.html.

29. **Larman, C.** *UML y patrones: introducción al análisis y diseño orientado a objetos.* s.l. : Félix Varela, 2004. ISBN: 8420534382.

30. **Mora, Roberto Canales.** AdictosAlTrabajo.com. *Patrones de GRASP.* [En línea] Autentia S.L. [Citado el: 21 de abril de 2015.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp.

31. **Juan, Francisco Javier Martínez.** *Guía de Construcción de Software en Java con patrones de diseño.*

32. **Ivar Jacobson, Grady Booch y James Rumbaugh.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* 1era Edición.Madrid : Addison Wesley, 2000. 84-7829-036-2.

Anexo IV: Pruebas unitarias

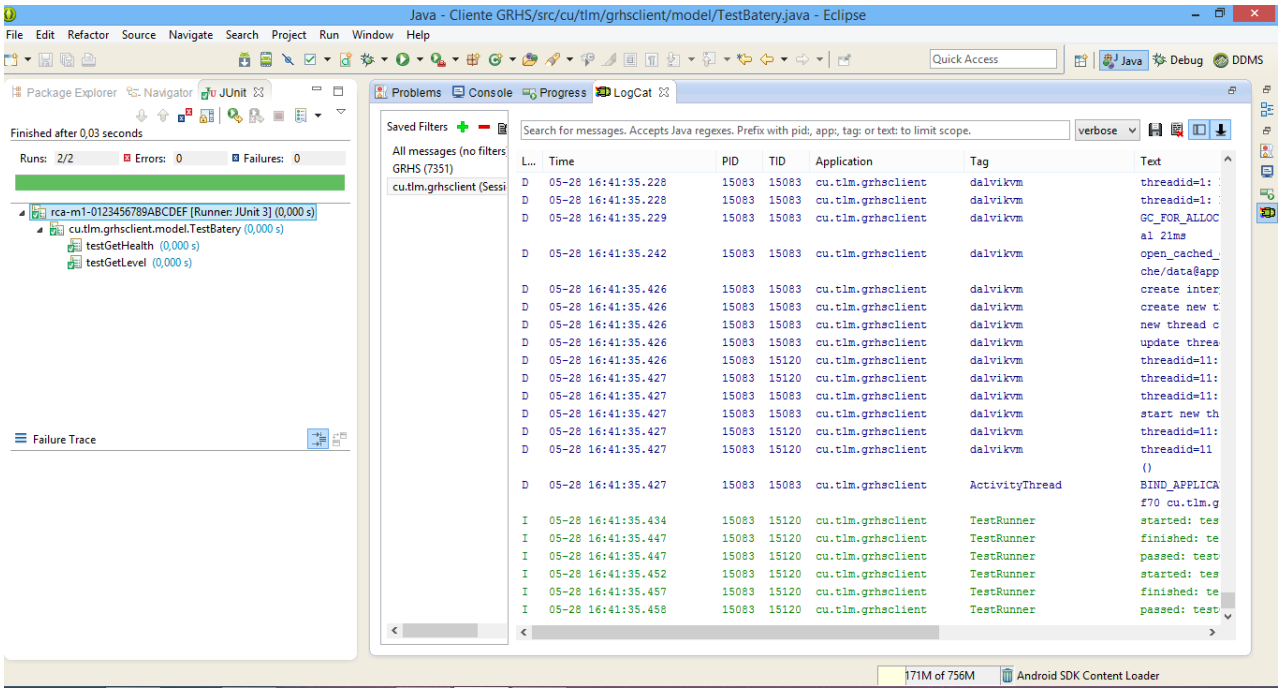


Figura 15: Prueba unitaria a la clase Battery.

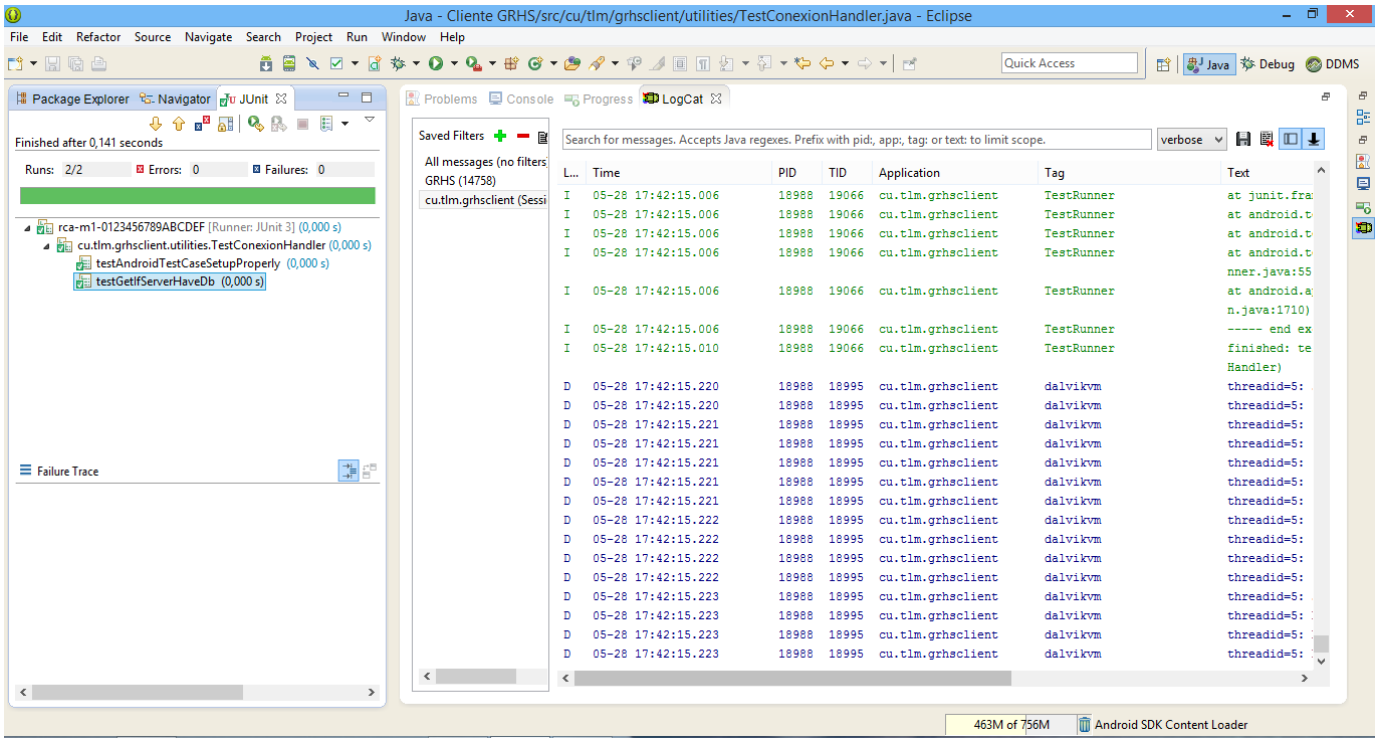


Figura 16: Prueba unitaria a la clase ConexionHandler.

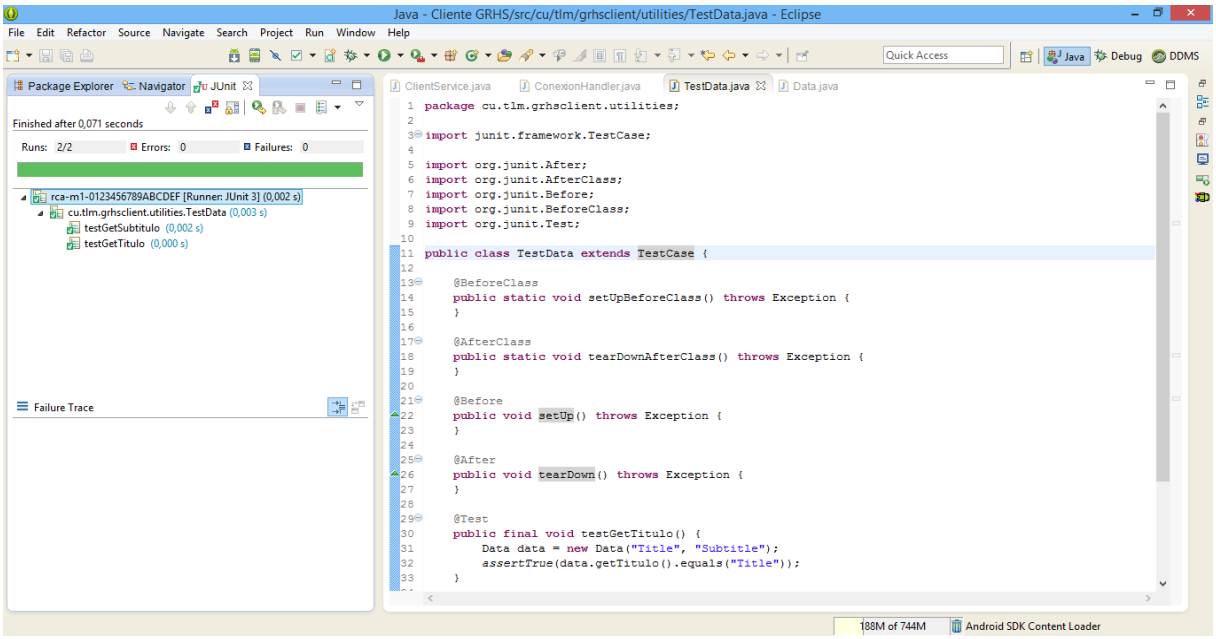


Figura 17: Prueba unitaria a la clase Data.

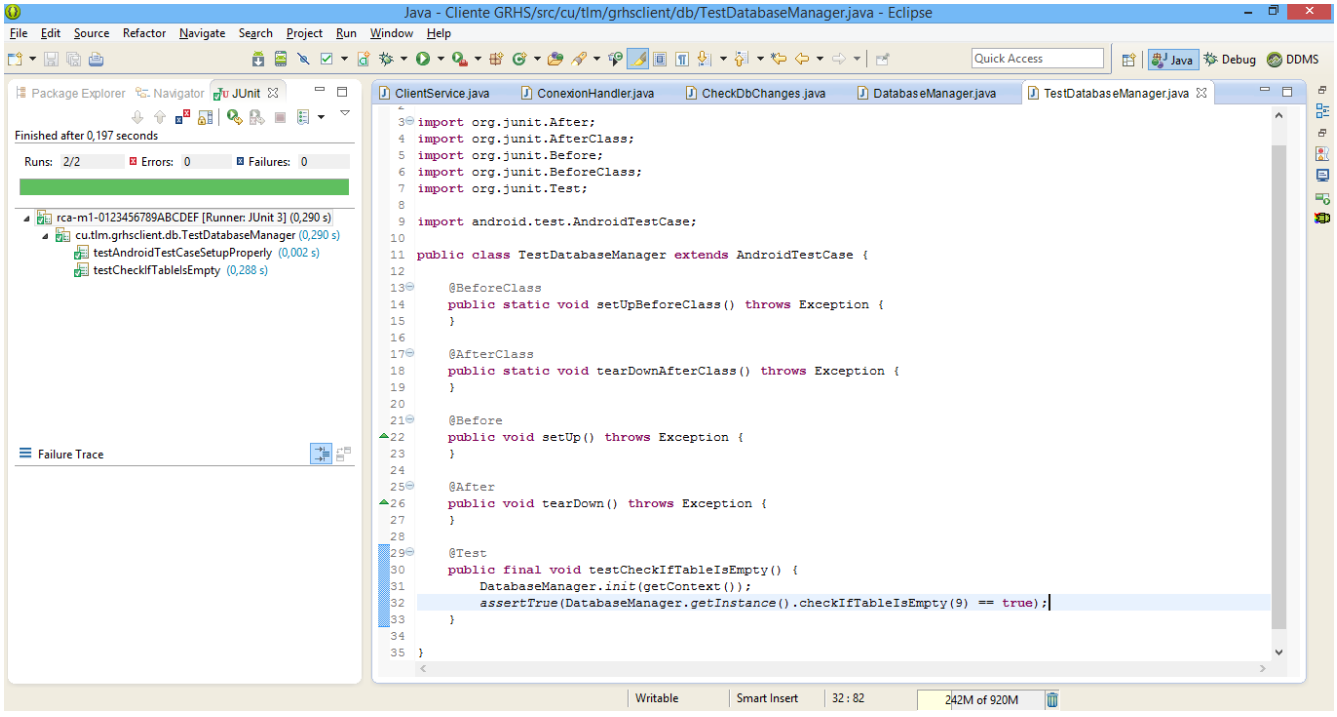


Figura 18: Prueba unitaria a la clase DatabaseManger al método CheckIfTableIsEmpty.

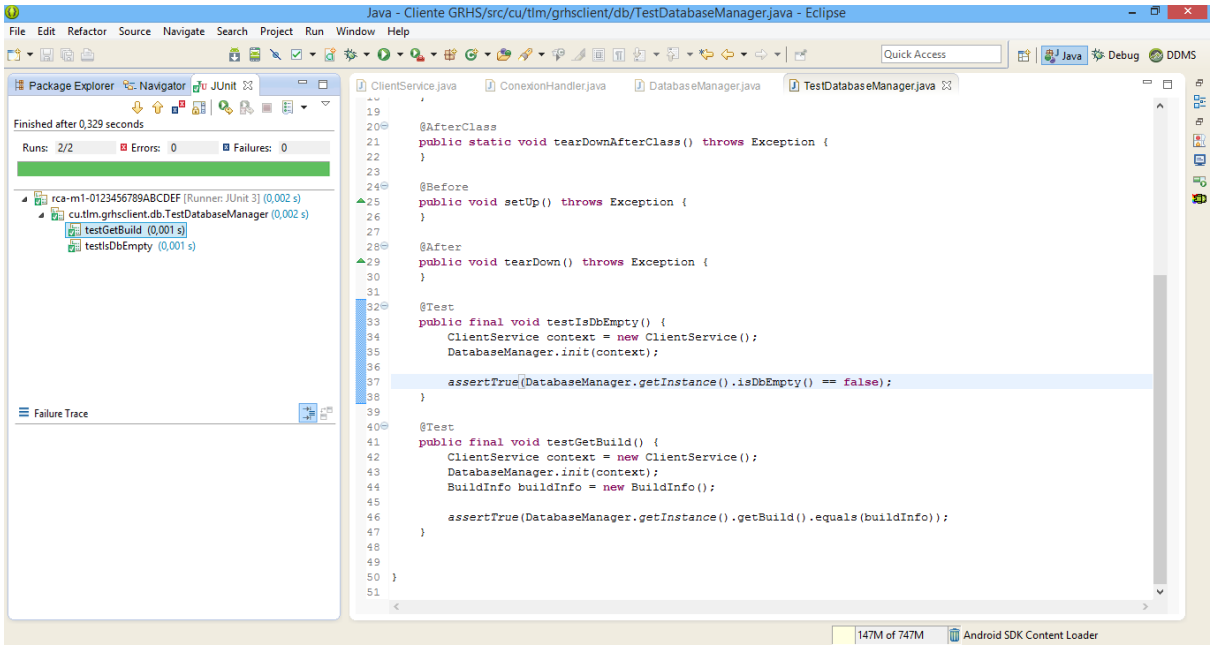


Figura 19: Prueba unitaria a la clase DatabaseManager al método DbEmpty.

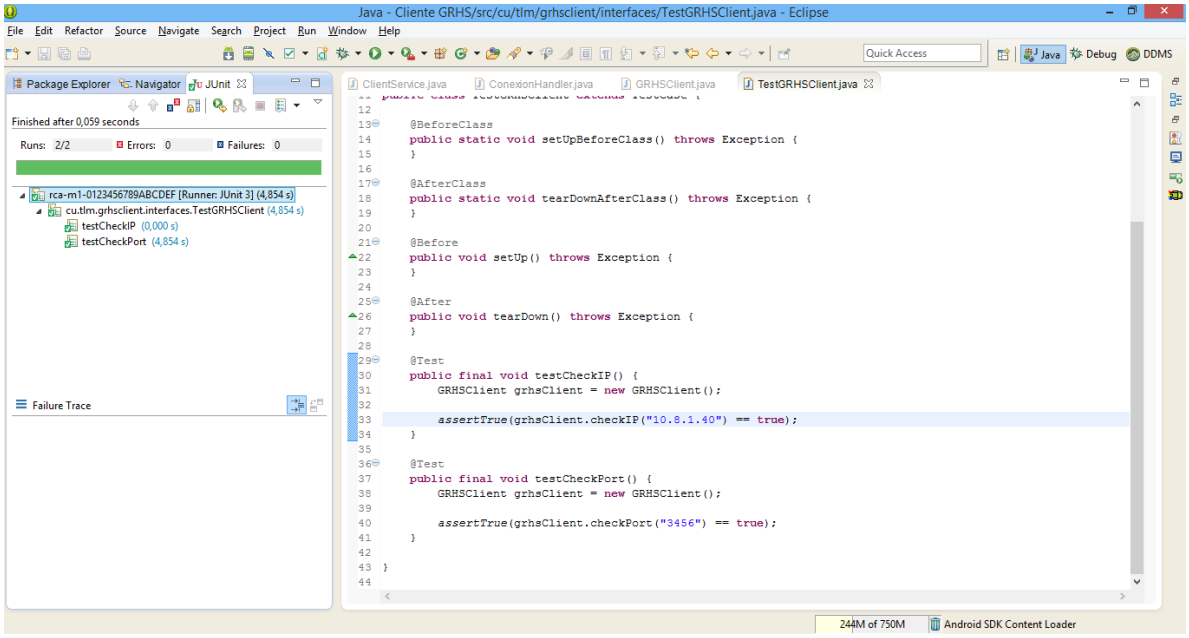


Figura 20: Prueba unitaria a la clase GRHSClient.