



Universidad de las Ciencias
Informáticas

Facultad 2

Trabajo de Diploma

**Presentado para optar por el título de Ingeniero en Ciencias
Informáticas**

Título

Aplicación Android para la administración del sistema Gestor de Recursos de
Hardware y Software

Autores

Julia Roxana García Andalia
Néstor Antonio Castro Castillo

Tutores

MSc. Ramón Alexander Anglada Martínez
Ing. Jenny De la Rosa Pasteur

La Habana, Cuba
Junio de 2016

Declaración de autoría

Los firmantes declaramos que somos los únicos autores del presente trabajo, que el contenido es de nuestra propia elaboración a excepción de donde se especifica de forma explícita en el texto y que el mismo no ha sido presentado para ninguna otra titulación o cualificación profesional, excepto como se especifica. Por la presente los firmantes autorizamos al Centro de Telemática de la Universidad de las Ciencias Informáticas a hacer uso de este trabajo en su beneficio.

Para que así conste, firmamos la presente a los 30 días del mes de Junio del año 2016.

Julia Roxana García Andalia

Nestor Antonio Castro Castillo

Firma del autor

Firma del autor

Dedicatoria

Dedicado a nuestros familiares y amigos, por la comprensión, afecto y apoyo incondicional que nos han brindado a lo largo de este largo y difícil trayecto que es la formación académica y profesional. A ustedes, que nos han acompañado en nuestra lucha por alcanzar nuestras metas y sueños.

Agradecimientos

Los autores desean hacer constar su agradecimiento a las personas que de una forma u otra han contribuido a la realización del presente trabajo; en particular, al Master en Ciencias Ramón Alexander Anglada Martínez y a la Ingeniera en Ciencias Informáticas Jenny de la Rosa Pasteur, quienes, en su rol de tutores del mismo, han sido una valiosa fuente de conocimientos, experiencia y apoyo técnico y moral.

Agradecimientos especiales al equipo de desarrollo del proyecto GRHS por la colaboración brindada en la aclaración de cuestiones técnicas asociadas a dicho sistema; en particular, al Ingeniero en Ciencias Informáticas Adrian Ezequiel Mena Rodríguez, por la capacitación ofrecida a los autores del presente trabajo en el despliegue de los servidores de prueba.

Finalmente, agradecer al Ingeniero en Ciencias Informáticas Pável Reyes Estévez por su colaboración en la revisión de los aspectos metodológicos de la investigación y al Master Néstor Antonio Castro Santiesteban por su colaboración en la revisión del Capítulo 1.

Resumen

La aparición de dispositivos móviles inteligentes, tales como los smartphones y los tablets, ha posibilitado el acceso a Internet de un mayor número de personas, debido a su menor costo con respecto a los ordenadores tradicionales. Como resultado, numerosas soluciones informáticas actuales se han integrado a las plataformas móviles en pos de aumentar su competitividad en el mercado de aplicaciones. Sin embargo, aún existen muchos sistemas que no han explorado las posibilidades de estas tecnologías.

El presente ejercicio académico aborda los problemas asociados a la no existencia de una interfaz de usuario, compatible con dispositivos móviles, para el sistema Gestor de Recursos de Hardware y Software desarrollado en la Universidad de las Ciencias Informáticas. La primera parte es un estudio de las formas en que sistemas similares implementan sus interfaces de usuario, teniendo en cuenta la compatibilidad o no de las mismas con dispositivos móviles. A continuación, los diferentes enfoques que existen en el desarrollo de aplicaciones móviles, y las ventajas y desventajas de cada uno, son analizados. Más adelante, se explica el diseño de la propuesta de solución a los problemas planteados. Finalmente se ofrecen resultados relevantes sobre el proceso de implementación y validación de la solución.

Palabras clave: dispositivos móviles, interfaz de usuario, compatibilidad, aplicaciones móviles.

Abstract

The emergence of smart mobile devices, such as smartphones and tablets, has enabled Internet access to a greater number of people, due to their lower cost as compared to traditional computers. As a result, many current software solutions are integrated into mobile platforms towards achieving a more competitive place in the applications market. However, there are still many systems that have not explored the possibilities of these technologies.

This academic exercise addresses the problems associated with the absence of a mobile device compatible user interface, for the Hardware and Software Resources Manager system developed at the University of Informatic Sciences. The first part is a study of the way in which similar systems implement their user interfaces, taking into account the compatibility or otherwise of these with mobile devices. Then, the different approaches that exist in developing mobile applications, and the advantages and disadvantages of each, are analyzed. Later, the design of the proposed solution to the problems discussed is explained. Finally, relevant results on the process of implementation and validation of the solution are offered.

Keywords: mobile devices, user interface, compatibility, mobile applications.

Índice

Introducción.....	11
Capítulo 1: Fundamentación teórica.....	15
1.1 Introducción.....	15
1.2 Sistema de gestión de recursos de hardware y software.....	15
1.3 Herramientas de gestión de recursos de hardware y software.....	15
1.3.1 Herramientas comerciales.....	15
1.3.2 Herramientas libres.....	16
1.4 Enfoques de desarrollo de aplicaciones móviles.....	18
1.4.1 Aplicación web móvil vs aplicación nativa.....	18
1.5 Metodología de desarrollo de software.....	19
1.5.1 SCRUM.....	20
1.5.2 AUP.....	20
1.5.3 Metodología XP.....	21
1.6 Herramientas de desarrollo.....	22
1.6.1 Open JDK 7.....	22
1.6.2 Android Studio 1.3.2.....	22
1.6.3 Android SDK Tools 23.0.5.....	23
1.6.4 Python 2.7.....	23
1.6.5 Django 1.4.....	24
1.6.7 Django REST Framework 2.2.1.....	24
1.7 Conclusiones del capítulo.....	24
Capítulo 2: Planificación y diseño del sistema.....	26
2.1 Introducción.....	26
2.2 Modelo de la propuesta de solución.....	26
2.2.1 Sincronización de datos.....	27
2.2.2 Autenticación.....	28
2.2.3 Consulta y manipulación de datos.....	28
2.2.4 Gestión de almacenamiento de datos en el dispositivo.....	29
2.3 Planeación.....	30
2.3.1 Historias de usuario.....	30

2.3.2 Características no funcionales del sistema	31
2.3.3 Tiempo de ejecución del proyecto	32
2.3.4 Iteraciones.....	33
2.3.5 Plan de entrega	34
2.4 Diseño de la propuesta de solución.....	35
2.4.1 Arquitectura de software.....	35
2.4.2 Patrones de diseño.....	36
2.4.3 Tarjetas Clase-Responsabilidad-Colaborador	39
Conclusiones del capítulo	40
Capítulo 3: Implementación y pruebas del sistema.....	41
3.1 Introducción	41
3.2 Estándares de codificación	41
3.2.1 Nomenclatura de las clases	41
3.2.2 Nomenclatura de los métodos y variables en el código de Gadmin Android.....	42
3.2.3 Nomenclatura de las constantes	42
3.2.4 Nomenclatura de los métodos y variables en el código del API REST de GRHS	43
3.3 Diagrama de despliegue	43
3.4 Tareas de ingeniería.....	44
3.6 Pruebas unitarias.....	45
3.7 Pruebas de aceptación	46
Conclusiones del capítulo	47
Conclusiones generales	48
Referencias bibliográficas	49
Bibliografía	52
Anexos	55
Anexo I Historias de Usuario.....	55
Anexo II Tarjetas CRC	60
Anexo III Tareas de Ingeniería.....	62
Anexo IV Pruebas de aceptación.....	69

Índice de figuras

Figura 1 Interfaz de administración web adaptable de OCS Inventory NG.	17
Figura 2 Interfaz de administración web adaptable de GLPI.....	17
Figura 3 Modelo de la propuesta de solución.	26
Figura 4 Estructura centrada en un Sync Adapter.	27
Figura 5 Estructura lógica de un Content Provider.	28
Figura 6 Secuencia de verificación de espacio disponible en el dispositivo móvil durante una sincronización.	29
Figura 7 Arquitectura Cliente/Servidor desglosada en sus dos niveles.....	35
Figura 8 Solicitud de creación de una nueva cuenta.	36
Figura 9 Relación de composición entre las clases SyncAdapter y SyncService.....	37
Figura 10 Uso de la clase ContentProviderOperations para realizar operaciones sobre la capa de acceso a datos persistentes de Gadmin Android.....	38
Figura 11 Esquema de aplicación del patrón Maestro/Esclavo en Gadmin Android.	39
Figura 12 Declaración de una clase en el código fuente del API REST de GRHS.....	42
Figura 13 Declaración de una clase en el código fuente de Gadmin Android.	42
Figura 14 Declaración de métodos y variables en el código fuente de Gadmin Android.	42
Figura 15 Declaración de una constante en el código fuente de Gadmin Android.	42
Figura 16 Declaración de métodos y variables en el código fuente del API REST de GRHS.....	43
Figura 17. Diagrama de despliegue para la propuesta de solución.	43
Figura 18 Clase de pruebas para comprobar el funcionamiento del servicio de autenticación del API REST de GRHS mediante el framework PyUnit.	45
Figura 19 Resultados de las pruebas de aceptación.	47

Índice de tablas

Tabla 1 Una explicación de cómo las aplicaciones nativas y web móviles presumiblemente resuelven características clave del desarrollo y uso de aplicaciones.....	18
Tabla 2 HU-1 Implementar un API REST en el servidor de GRHS.....	30
Tabla 3 Estimación del costo en tiempo por historia de usuario.....	33
Tabla 4 Plan estimado de duración de las iteraciones.....	34
Tabla 5 Plan de entrega por iteraciones.....	34
Tabla 6. Tarjeta CRC AccountAuthenticator.....	40
Tabla 7 TI Implementar el servicio GetHostData.....	44
Tabla 8 PA Consultar servicios del API REST.....	46
Tabla 9 HU-2 Implementar un mecanismo de manejo de permisos para el acceso al API REST ..	55
Tabla 10 HU-3 Implementar un mecanismo de gestión de cuenta de usuario.....	55
Tabla 11 HU-4 Implementar un mecanismo de descarga y sincronización de datos a demanda...	56
Tabla 12 HU-5 Mostrar estadísticas del sistema.....	57
Tabla 13 HU-6 Mostrar información de agentes, inventario, acciones, incidencias.....	57
Tabla 14 HU-7 Implementar una interfaz para el manejo de preferencias de la aplicación.....	58
Tabla 15 HU-8 Gestionar la conexión de la aplicación mediante el protocolo HTTPS.....	59
Tabla 16 HU-9 Implementar un mecanismo de búsqueda.....	59
Tabla 17 Tarjeta CRC AuthenticationActivity.....	60
Tabla 18 Tarjeta CRC AuthenticationService.....	60
Tabla 19 Tarjeta CRC DataBaseContentProvider.....	61
Tabla 20 Tarjeta CRC DataBaseHelper.....	61
Tabla 21 Tarjeta CRC DataBaseContract.....	61
Tabla 22 Tarjeta CRC SyncAdapter.....	62
Tabla 23 Tarjeta CRC SyncService.....	62
Tabla 24 TI Implementar el servicio GetHostByld.....	62
Tabla 25 TI Implementar el servicio GetUnifiedMB.....	63
Tabla 26 TI Implementar el servicio GetLocations.....	63
Tabla 27 TI Implementar el servicio GetIncidences.....	63
Tabla 28 TI Implementar el servicio GetStats.....	63
Tabla 29 TI Implementar el servicio GetInactiveTime.....	64
Tabla 30 TI Implementar el servicio GetAlarms.....	64
Tabla 31 TI Implementar filtros de ubicaciones.....	64
Tabla 32 TI Integrar el sistema de autenticación del API REST con los servidores LDAP.....	65
Tabla 33 TI Implementar un Account Authenticator.....	65
Tabla 34 TI Implementar un Content Provider.....	66
Tabla 35 TI Implementar un Sync Adapter.....	66
Tabla 36 TI Implementar vistas de estadísticas del sistema.....	66

Tabla 37 TI Implementar vistas de listados de objetos por cada categoría	67
Tabla 38 TI Implementar vistas de detalles de cada objeto	67
Tabla 39 TI Implementar la actividad de preferencias	67
Tabla 40 TI Modificar las clases que manejan la conexión al servidor para que empleen el protocolo de transferencia HTTPS	68
Tabla 41 TI Implementar un mecanismo de búsqueda	68
Tabla 42 PA Verificar el funcionamiento del mecanismo de manejo de permisos para el acceso al API REST	69
Tabla 43 PA Crear una cuenta de usuario nueva	69
Tabla 44 PA Sincronización manual	70
Tabla 45 PA Mostrar estadísticas del sistema	70
Tabla 46 PA Mostrar información de agentes, inventario, acciones, incidencias	71
Tabla 47 PA Configurar las preferencias de la aplicación	71
Tabla 48 PA Realizar una solicitud de autenticación al servidor mediante una conexión HTTPS ..	72
Tabla 49 PA Ejecutar una búsqueda filtrada.....	72

Introducción

En la actualidad, a nivel mundial, se ha generalizado el uso de dispositivos móviles para acceder a una gran variedad de recursos en Internet. Cada vez está más expandido el uso de estos dispositivos, por las facilidades que ofrece y por su menor costo en comparación con los ordenadores tradicionales de escritorio o portátiles. Esta realidad se desenvuelve aparejada al desarrollo de las tecnologías de conexión inalámbricas, particularmente la tecnología Wi-Fi¹.

Wi-Fi es el medio por defecto para la conectividad de hoy en día. Lleva aproximadamente la mitad de todo el tráfico de Internet y 10 veces más tráfico que el celular. (1)

La Universidad de las Ciencias Informáticas (UCI) es una institución estatal que cuenta con Puntos de Acceso (AP) Wi-Fi dispuestos en las áreas principales de la misma, tales como: salones de reunión, decanatos, algunos puntos del complejo residencial, entre otros. Dicha red permite el acceso de estudiantes y trabajadores a los diferentes servicios, portales web y sistemas de gestión disponibles en la red de la institución, así como el acceso controlado a Internet. El Gestor de Recursos de Hardware y Software (GRHS), es uno de tales sistemas. En el Encuentro Nacional de Tecnologías de la Información de 2015 (2), se describió a GRHS como una solución informática que permite realizar el inventario de hardware y software de las computadoras donde se encuentre instalado, así como la detección de incidencias que se puedan producir en las mismas, alertando a los interesados. Además, genera estadísticas que contribuyen a la toma de decisiones, por parte de los especialistas, con respecto al seguimiento de las políticas de seguridad informática.

Actualmente acceden a GRHS los asesores de seguridad informática de la UCI, con el objetivo de conocer la información de los recursos de hardware y software que tienen asignados, verificar el estado del cumplimiento de las políticas de seguridad informática, realizar reportes para tomar decisiones administrativas, entre otras tareas. A partir de la instalación de los AP en la universidad, se ha hecho posible acceder a dicho sistema desde dispositivos móviles.

GRHS consta de tres aplicaciones: el colector de inventarios, **gclient**, el centralizador de inventarios, **gserver**, y la consola de administración, **gadmin**. La aplicación **gadmin** permite consultar la información de los inventarios y realizar las configuraciones del sistema. Actualmente no cuenta con una interfaz orientada a la visualización de los datos de GRHS en dispositivos móviles. Al ejecutar la consola de administración desde el navegador web de tales dispositivos, la interfaz se deforma debido a que no se construyó para adaptarse a los tamaños de pantalla reducidos que caracterizan a este tipo de equipamiento tecnológico. Lo anterior obliga al usuario a realizar constantes cambios en las dimensiones del área visible de la pantalla del navegador para poder entender la información que se muestra.

¹ **Wi-Fi:** Wireless Fidelity, un sistema para la conexión de equipos electrónicos tales como ordenadores y agendas electrónicas a internet sin necesidad de cables.

Otro aspecto importante es que el rendimiento de **gadmin**, durante su ejecución en el navegador web de un dispositivo móvil, es bastante más bajo que lo deseable para un producto de su tipo, lo cual afecta directamente a la experiencia de usuario y, por consiguiente, a la calidad del software.

El no contar con una interfaz para dispositivos móviles, limita la competitividad de GRHS como un potencial producto a ser adquirido por instituciones que cuenten con la infraestructura necesaria para su explotación. Teniendo en cuenta que el éxito de las aplicaciones móviles ha supuesto un nuevo empuje para el mercado de los tablets² y smartphones³, cada vez más valorados por las bondades de sus sistemas operativos, sería lógico asumir que contar con una aplicación para esta plataforma tecnológica puede expandir considerablemente las posibilidades comerciales de cualquier solución informática moderna; potencial que actualmente no es posible explotar en GRHS.

Lo anteriormente planteado da paso al siguiente **problema a resolver**: *La consola de administración del sistema GRHS no cuenta con una interfaz visual orientada a dispositivos móviles, lo cual trae como consecuencia limitaciones en su explotación y facilidades de uso.*

El **objeto de estudio** de esta investigación está centrado en: *Las técnicas de desarrollo de aplicaciones para dispositivos móviles.*

El **objetivo general** a cumplir para darle solución al problema planteado es: *Desarrollar una aplicación Android para la administración del sistema Gestor de Recursos de Hardware y Software.*

El **campo de acción** se enmarca en: *El desarrollo de aplicaciones de gestión para dispositivos móviles con sistema operativo Android.*

Para este fin se proponen como **tareas** que se utilizarán de guía y apoyo para solucionar el problema identificado:

1. Elaboración del marco teórico de la investigación, para definir los elementos fundamentales que regirán el desarrollo de la solución propuesta.
2. Realización del estudio del estado del arte con el objetivo de evaluar las soluciones existentes a problemas similares al que se trata en la investigación, así como comparar los diferentes enfoques en el desarrollo de aplicaciones móviles.
3. Evaluación de la metodología de desarrollo de software a utilizar para la descripción de los artefactos que se generan durante el desarrollo de la aplicación.
4. Estudio de las características del sistema GRHS, para lograr la integración del mismo con la aplicación que se pretende desarrollar.

² **Tablet**: también conocido como computador tablet, es una pequeña computadora de una pieza, que el usuario opera mediante una pantalla táctil.

³ **Smartphone**: un teléfono móvil que puede ser utilizado como una computadora pequeña con conectividad a Internet.

5. Definición de las características que debe tener la aplicación de administración, para determinar las funcionalidades a implementar en la solución propuesta.
6. Selección de las tecnologías y herramientas necesarias para el desarrollo de la solución propuesta.
7. Implementación de la aplicación de administración.
8. Selección de los tipos de pruebas necesarias para comprobar que la aplicación implementada cumple con los objetivos propuestos.

Métodos teóricos:

El método **analítico-sintético**: se utilizó para simplificar todas las citas, apuntes y datos tomados al respecto del tema tratado en la presente investigación. La revisión bibliográfica se realizó sobre un conjunto de libros, publicaciones y documentos en soporte electrónico, que se encuentran situados en páginas web, artículos científicos, entre otros.

El método **histórico-lógico**: se utilizó para el trabajo recopilatorio sobre las características de los sistemas de gestión de recursos de hardware y software, los diferentes paradigmas en el desarrollo de aplicaciones móviles y los diferentes aspectos que conectan a ambos.

El método **modelación**: permitió realizar las actividades encaminadas a la adaptación o construcción de modelos a través de los cuales se representan parcial o totalmente las cualidades esenciales del fenómeno objeto de estudio.

Métodos empíricos:

El método **observación**: se utilizó para obtener de forma directa, la información de la realidad objetiva del comportamiento de la gestión de los recursos de hardware y software que se desarrolla en el sistema GRHS de la Universidad de las Ciencias Informáticas.

El presente ejercicio académico consta de la siguiente estructura:

El **Capítulo 1: *Fundamentación teórica***, incluye los aspectos teóricos que soportan esta investigación y el estudio de las tendencias y paradigmas actuales, metodología de desarrollo, las herramientas y las tecnologías propuestas para el desarrollo de la aplicación de administración del sistema GRHS para dispositivos móviles con sistema operativo Android (esta aplicación será referida en el contenido del presente trabajo como Gadmin Android).

El **Capítulo 2: *Planificación y diseño del sistema***, en este capítulo se describen las características de la aplicación a desarrollar. Se realizan las especificaciones de las funcionalidades del sistema y se elabora el plan de entregas por iteraciones.

El **Capítulo 3: *Implementación y pruebas del sistema***, en este capítulo se plantea la construcción de la solución propuesta en el Capítulo 2. Se presentan las principales tareas de ingeniería a realizar y los resultados de las pruebas realizadas al sistema, para comprobar la efectividad de la aplicación.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el siguiente capítulo se realiza un estudio de las diferentes formas en que los sistemas de gestión de recursos de hardware y software existentes implementan sus interfaces de administración, teniendo como elemento fundamental de interés la compatibilidad de las mismas con dispositivos móviles, así como las ventajas que ofrece la integración de estos sistemas con las tecnologías móviles. Se ofrece un estudio de los principales enfoques de desarrollo de aplicaciones móviles; además, se detallan las características principales de la metodología y las herramientas de desarrollo a utilizar durante la implementación de la aplicación.

1.2 Sistema de gestión de recursos de hardware y software

Para una mayor comprensión de los temas tratados en la siguiente investigación, se ofrecerá una definición al concepto clave **sistema de gestión de recursos de hardware y software**, considerando que es un conjunto de herramientas elaboradas con el objetivo de supervisar los equipos de cómputo conectados a una red de ordenadores y que facilita el inventario de los recursos de hardware y software instalados en los mismos.

1.3 Herramientas de gestión de recursos de hardware y software

1.3.1 Herramientas comerciales

Total Network Inventory

Total Network Inventory es una solución de inventario de software y auditoría de PC⁴. Soporta las plataformas GNU/Linux, Mac OS y Windows. Examina todos los ordenadores y portátiles en la red y genera la información sobre el sistema operativo, dispositivos y software instalado. No es necesaria la instalación previa de agentes en las computadoras remotas o ninguna otra preparación para una auditoría de red.

La administración de Total Network Inventory se realiza desde una aplicación de escritorio desarrollada para la plataforma Windows de Microsoft. De acuerdo con la documentación provista por Softinventive Lab Inc. (3), la unidad de administración del sistema requiere Windows XP Professional SP3 o Windows Server 2003 SP2 como sistema operativo para su instalación y ejecución. Esto demuestra una clara limitación del alcance de esta solución, en lo que a la administración de la misma se refiere, hacia la plataforma Windows de Microsoft.

⁴ **PC:** Personal Computer (Computadora Personal).

NetSupport DNA

NetSupport DNA es una solución completa de gestión de activos de TI⁵ y escritorios. Ofrece una gran cantidad de componentes que incluyen: la detección automática de dispositivos; inventario de hardware y software, gestión de usuarios y licencias de software.

La administración del sistema NetSupport DNA se realiza desde las diferentes consolas desarrolladas a este propósito, las cuales están disponibles en forma de aplicaciones de escritorio y aplicaciones para dispositivos móviles. La consola de administración móvil es una aplicación compatible con los sistemas operativos móviles iOS y Android.

La app móvil de DNA permite a un técnico, cuando está delante de su ordenador, buscar y ver un inventario detallado de hardware y software de cualquier PC de la red de la empresa. La app móvil también incluye un escáner de código QR para ayudar a identificar de forma instantánea cualquier PC, bien desde un código QR mostrado en la pantalla por DNA, o de una etiqueta fijada al dispositivo. NetSupport DNA proporciona también una función de creación de etiquetas con códigos QR que incluye compatibilidad con detalles personalizados. La aplicación muestra un historial de todos los cambios efectuados al hardware, así como todas las instalaciones o desinstalaciones de software. (4)

1.3.2 Herramientas libres

OCS Inventory NG

OCS Inventory Next Generation es una herramienta libre de inventario de activos de TI. Algunas de las características que posee incluyen: la posibilidad de realizar reportes de los inventarios, soporte a una amplia gama de plataformas entre las que se destacan Windows y GNU/Linux, servicios web accesibles a través de interfaces SOAP⁶ y soporte a la adición de complementos o plugins⁷ a través de un API⁸. Cuenta con una consola de administración accesible a través de una interfaz web responsive⁹, la cual puede ponerse a prueba desde el sitio oficial del proyecto (5). La siguiente figura ilustra dicha interfaz en pantallas de diferentes dimensiones y orientación.

⁵ **TI:** Tecnologías de la Información.

⁶ **SOAP:** Simple Object Access Protocol (Protocolo de Acceso de Objeto Simple), protocolo estándar que define cómo dos objetos en diferentes procesos deben comunicarse.

⁷ **Plugin:** componente de software o programa que incrementa las funcionalidades de un programa principal.

⁸ **API:** Application Programming Interface (Interfaz de Programación de Aplicaciones), es un conjunto de herramientas de código, que ofrece cierta biblioteca de software, para ser usada por un sistema externo como capa de abstracción.

⁹ **Responsive:** sensible, adaptativo. Es la construcción de sitios web capaces de adaptar su diseño, contenido y apariencia para optimizar la experiencia del usuario a través de dispositivos de diferentes tamaños y capacidades, incluyendo teléfonos inteligentes, tabletas y ordenadores de pantalla ancha.

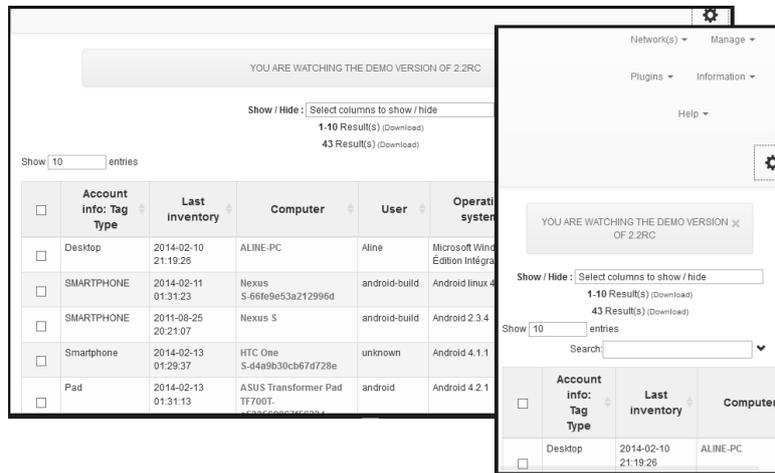


Figura 1 Interfaz de administración web adaptable de OCS Inventory NG.

La interfaz de OCS Inventory es bastante sencilla lo cual deriva en un tiempo de respuesta rápido, característica deseable para los dispositivos móviles ya que usualmente tienen menos prestaciones que los ordenadores tradicionales. Sus características de diseño adaptable hacen posible su uso desde el navegador de un dispositivo móvil, lo cual expande sus posibilidades de explotación considerablemente.

Gestión Libre de Parque Informático (GLPI)

GLPI es una aplicación de código abierto, muy útil en empresas con varias sedes, para gestionar el inventario. Es la interfaz de administración de un sistema de inventario, ya que toma sus datos de otras herramientas como OCS Inventory NG y Fusion Inventory. Incluye datos administrativos como pueden ser: períodos de validez de las licencias de software, garantías y tiempos de caducidad de las mismas, o datos de ubicación de los equipos. Cuenta con una consola de administración web responsive que puede ser puesta a prueba desde el sitio oficial del proyecto (6). La siguiente figura corresponde a la visualización de dicha interfaz en diferentes tamaños de pantalla.

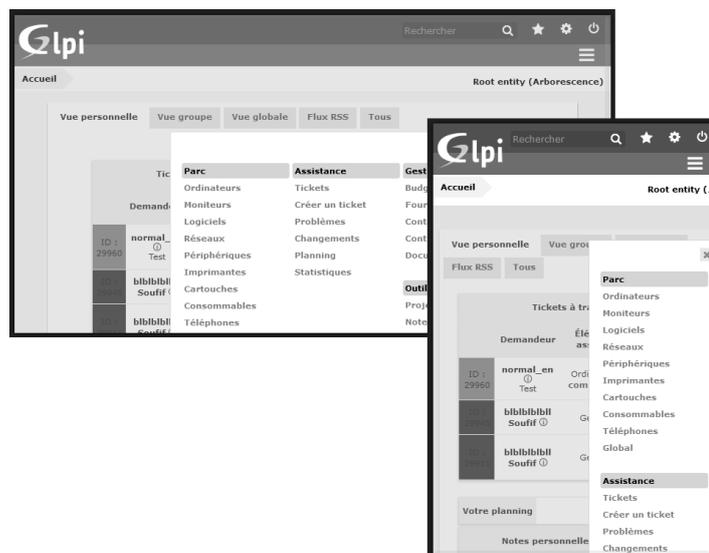


Figura 2 Interfaz de administración web adaptable de GLPI.

En su última versión estable, GLPI 0.90.1, se incluyeron mejoras considerables en la interfaz, centradas en el contraste de los elementos y en una pantalla más sensible para dispositivos móviles tales como tablets y smartphones (7).

1.4 Enfoques de desarrollo de aplicaciones móviles

Sawyer Bateman (8) plantea que siempre ha existido una división entre las aplicaciones móviles nativas (aquellas construidas para ejecutarse en una plataforma específica) y las no nativas (desarrolladas para ejecutarse en un navegador bajo las pautas del diseño adaptable o responsive). En sus principios, la funcionalidad de las aplicaciones nativas era mucho más limitada de lo que es actualmente, de modo que la brecha no era tan notable; sin embargo, con los avances en hardware, servicios basados en la ubicación, video, voz, entre otros, la brecha entre lo que pueden hacer un sitio web móvil en contraste con una aplicación nativa, es cada vez más evidente, creando una tensión considerable entre el diseño responsive y el diseño nativo.

Es de suma importancia la correcta elección entre ambos enfoques a la hora de desarrollar una aplicación para dispositivos móviles. A continuación se ofrece un estudio comparativo entre los paradigmas de desarrollo: aplicación web móvil y aplicación nativa, con el objetivo de optar por la variante que contribuya en mayor medida a la solución del problema planteado en la presente investigación.

1.4.1 Aplicación web móvil vs aplicación nativa

Existe un conjunto de aspectos generales, de desarrollo y uso de aplicaciones, que se considera que cada tipo de aplicación móvil (web o nativa) puede resolver con mayor o menor efectividad. William Jobe (9) desglosa estos aspectos en la tabla que se muestra a continuación:

Tabla 1 Una explicación de cómo las aplicaciones nativas y web móviles presumiblemente resuelven características clave del desarrollo y uso de aplicaciones.

Característica	Aplicación Nativa	Aplicación Web
Creación vs consumo de contenido	Son más adecuadas para la creación de contenido debido al acceso al hardware y al rendimiento.	Son menos adecuadas para la creación de contenido pero igualmente idóneas para el consumo.
Experiencia de usuario	Integración perfecta con el sistema operativo.	Integración limitada, requiere marcos de trabajo externos.
Frecuencia de actualización	Formales a través de las tiendas de aplicaciones.	Más informales y equivalentes a actualizaciones de sitios web.
Rendimiento	Rendimiento máximo y acceso al hardware del dispositivo.	El desempeño es dependiente de JavaScript y de los exploradores

		web móviles; acceso limitado al hardware del dispositivo.
Funcionalidad	Todas las funcionalidades del sistema operativo móvil están disponibles.	La mayoría de las funcionalidades del sistema operativo están disponibles.
Desarrollo	Requiere desarrollo específico para cada sistema operativo móvil.	Lenguajes y exploradores web abiertos hacen posible el desarrollo tipo "Escriba una vez, ejecute en todas partes".

Actualmente se considera, con un amplio margen, que desde un punto de vista puramente de diseño e interfaz de usuario, el enfoque nativo es substancialmente mejor. En lugar de intentar adaptarse al medio (el sistema operativo del dispositivo móvil), las aplicaciones nativas viven en el medio. (8)

Mientras que las aplicaciones que se ejecutan en el entorno del navegador web del dispositivo deben solicitar de los servidores todos los datos de generación de interfaz gráfica, hojas de estilo y código JavaScript, las aplicaciones nativas solo requieren obtener los datos en bruto en cada solicitud HTTP (por ejemplo, en forma de objetos JSON¹⁰), ya que la gestión de la interfaz gráfica se maneja dentro de la propia aplicación. Lo anterior implica que las aplicaciones nativas realicen en promedio menos consultas al servidor de aplicaciones que las responsive, lo cual resulta en un mejor tiempo de respuesta y menor consumo de los limitados recursos de hardware de los que disponen generalmente los dispositivos móviles.

1.5 Metodología de desarrollo de software

Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo de software. Letelier y Penadés (10) las dividen entre las propuestas más tradicionales, centradas especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán, y las centradas en dimensiones tales como el factor humano o el producto software. La segunda variante corresponde a la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software en iteraciones cortas. Este enfoque ha mostrado su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad.

Los equipos de desarrollo de aplicaciones empresariales utilizan prácticas tradicionales para definir y desarrollar aplicaciones de escritorio; sin embargo, la mayoría no funcionan con el desarrollo de

¹⁰ **JSON:** JavaScript Object Notation (Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

aplicaciones móviles, debido a la diversidad de dispositivos, conectividad de red y otras consideraciones específicas para móviles, (...) En cambio, los directores de desarrollo de aplicaciones deben utilizar pruebas funcionales, de rendimiento, de carga y de experiencia de usuario, así como las prácticas de desarrollo ágil. (11)

Van L. Baker, vicepresidente de investigaciones de Gartner Inc. (11), plantea que existen varias razones por las que las prácticas de desarrollo tradicionales no tienen éxito en el desarrollo de aplicaciones móviles, a pesar de que han funcionado históricamente. En primer lugar, las aplicaciones móviles son una nueva categoría para la mayoría de los usuarios y, en segundo lugar, están limitadas por la naturaleza de la plataforma y el tamaño de la pantalla. Lo anterior sugiere que el uso de metodologías ágiles es el enfoque práctico más favorable para el desarrollo de este tipo de aplicaciones; sin embargo, existen diversas opciones posibles a la hora de optar por una metodología entre las que pertenecen a este grupo. A continuación se ofrece una breve descripción de algunas de ellas.

1.5.1 SCRUM

SCRUM es una metodología especialmente desarrollada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir de la siguiente forma: el desarrollo de software se realiza mediante iteraciones denominadas sprints¹¹, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Al finalizar un sprint, las tareas que se han realizado, y con las que el cliente ha quedado conforme, no se vuelven a modificar. El equipo de desarrollo trata de seguir el orden de prioridad que marca el cliente pero puede modificarlo si determina que es mejor hacerlo y cada miembro del equipo trabaja de forma individual. Esta metodología se basa más en la administración del proyecto que en la propia programación o creación del producto.

1.5.2 AUP

El Proceso Unificado Ágil de Scott Ambler, o Agile Unified Process (AUP) en inglés, es una versión simplificada del Proceso Unificado Racional (RUP). Este describe, de una manera simple y fácil de entender, la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. AUP aplica un conjunto de técnicas ágiles entre las que se incluyen:

- Desarrollo Dirigido por Pruebas (Test Driven Development - TDD en inglés)
- Modelado ágil
- Gestión de cambios ágil

¹¹ **Sprint:** carrera corta, a toda velocidad.

- Refactorización de base de datos para mejorar la productividad

En la Universidad de las Ciencias Informáticas se ha realizado una variación de esta metodología de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la institución (12). En esta variación la metodología cuenta con 3 fases:

Inicio: se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se llevará a cabo el mismo.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase, el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.5.3 Metodología XP

XP¹² es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Para el desarrollo de Gadmin Android se escoge XP como metodología que guiará dicho proceso, teniendo en cuenta los aspectos que se listan a continuación:

- El proyecto es de corta duración y todo el trabajo de codificación es llevado a cabo por dos programadores trabajando en conjunto.
- No es necesaria la generación excesiva de artefactos y roles ya que el proyecto es pequeño y está diseñado para ser realizado en el menor tiempo posible.
- Los requisitos suelen cambiar con frecuencia en la medida en que avanza el desarrollo del proyecto, así el cliente puede ir añadiendo Historias de Usuario, dividir las para agilizar el

¹² **XP:** eXtreme Programming (Programación Extrema).

trabajo o eliminarlas simplemente. XP permite al equipo de trabajo modificar los planes de desarrollo conforme a lo anterior.

- Tanto cliente como desarrolladores forman parte del equipo de trabajo de forma tal que se logra una interacción constante entre ellos. Esto posibilita la retroalimentación, corrección de errores y finalmente la realización de un producto que cumpla las expectativas de las partes involucradas en el proceso.

1.6 Herramientas de desarrollo

En el presente epígrafe se explican las herramientas utilizadas en el desarrollo de la aplicación propuesta; así como algunas herramientas que se utilizan para el desarrollo del sistema GRHS en el centro de Telemática de la UCI. Estas últimas servirán para implementar las interfaces de comunicación entre GRHS y Gadmin Android.

1.6.1 Open JDK 7

Open JDK 7 es una implementación de código abierto de la séptima edición de la plataforma de desarrollo Java Standard Edition. Un conjunto de herramientas (programas y bibliotecas) que permiten desarrollar aplicaciones en el lenguaje de programación Java (13). Es la plataforma de desarrollo base de este proyecto debido a que Java es el lenguaje de programación estándar para el desarrollo de aplicaciones Android.

1.6.2 Android Studio 1.3.2

Android Studio es el IDE¹³ oficial para el desarrollo de aplicaciones Android. Entre sus capacidades, Android Studio ofrece:

- Un sistema de compilación flexible basado en Gradle¹⁴.
- Plantillas de código para asistir en la construcción de características comunes de aplicaciones.
- Editor de diseño con soporte para edición de temas tipo *arrastrar y soltar*.
- Herramientas Lint¹⁵ para capturar desempeño, usabilidad, compatibilidad entre versiones y otros problemas.

¹³ **IDE:** Integrated Development Environment (Entorno de Desarrollo Integrado), herramienta informática que proporciona servicios integrales para facilitar el desarrollo de software.

¹⁴ **Gradle:** herramienta utilizada para la automatización de proyectos en Java, y otros lenguajes, que facilita procesos de compilación y gestión de dependencias.

¹⁵ **Lint:** herramientas de software diseñadas para realizar tareas de verificación de código en cualquier lenguaje de programación.

1.6.3 Android SDK Tools 23.0.5

El SDK¹⁶ de Android provee las bibliotecas API y las herramientas de desarrollo necesarias para compilar, probar y depurar aplicaciones Android. Se utiliza esta versión del SDK debido a que es la más estable disponible en los repositorios de herramientas de software de la UCI en el momento en que se inicia la presente investigación.

Android API 21

Cada aplicación Android se desarrolla en base a una versión específica del framework¹⁷ API que ofrece el SDK, en pos de mantener la compatibilidad de la misma con versiones anteriores del sistema operativo. El nivel API es un valor entero que identifica de forma exclusiva la revisión del framework API que ofrece una versión de la plataforma Android (14). Las aplicaciones pueden utilizar el framework API para interactuar con el sistema Android subyacente. Cada revisión del API consiste en:

1. Un conjunto básico de paquetes y clases.
2. Un conjunto de elementos y atributos XML¹⁸ para declarar un archivo de manifiesto.
3. Un conjunto de elementos y atributos XML para declarar el acceso a los recursos.
4. Un conjunto de Intents¹⁹ para gestionar las acciones.
5. Un conjunto de permisos que las aplicaciones pueden solicitar, así como refuerzos de permisos incluidos en el sistema.

Cada versión sucesiva de la plataforma Android puede incluir actualizaciones del framework de aplicaciones que ofrece. La aplicación Android propuesta en el presente trabajo se implementará utilizando el framework API 21, lo cual implica que la compatibilidad de la misma se garantiza hasta la versión 5 de Android, Lollipop. Dicha versión es la más estable soportada en la revisión del SDK que se utilizará en el desarrollo de la aplicación.

1.6.4 Python 2.7

Python es un lenguaje de programación interpretado que incluye programación orientada a objetos y ofrece una manera sencilla de desarrollar programas con componentes reutilizables. Elimina preocupaciones referentes a detalles de bajo nivel, como manejar la memoria empleada por el

¹⁶ **SDK:** Software Development Kit (Paquete de Desarrollo de Software), es un conjunto de herramientas para el desarrollo de aplicaciones para una plataforma específica.

¹⁷ **Framework:** estructura conceptual y tecnológica de soporte, definido normalmente con artefactos o módulos concretos de software, que sirve de base para la organización y desarrollo de aplicaciones.

¹⁸ **XML:** eXtensible Markup Language, lenguaje de marcado desarrollado por el World Wide Web Consortium (W3C).

¹⁹ **Intent:** es un objeto de mensajería que se utiliza para solicitar acciones entre componentes de una aplicación Android.

programa, permitiendo a los desarrolladores centrarse más en el código que en la sintaxis. La versión 2.7 de Python es utilizada en el desarrollo del sistema GRHS y se usará en la implementación de las modificaciones necesarias en el código fuente de **gserver**, con el objetivo de lograr la interoperabilidad entre este y Gadmin Android.

1.6.5 Django 1.4

Django es un framework de aplicación de alto nivel, que fomenta el desarrollo rápido y el diseño limpio y pragmático. Se encarga de gran parte del desarrollo web, para que el desarrollador pueda centrarse en la escritura de su aplicación. Es gratuito y de código abierto, con una amplia comunidad de desarrollo y documentación detallada (15). La versión 1.4 de esta herramienta es la base utilizada en el desarrollo de GRHS y la principal dependencia a instalar para el desarrollo del API REST²⁰ que permitirá la comunicación entre **gserver** y Gadmin Android.

1.6.7 Django REST Framework 2.2.1

Django REST Framework es un conjunto de herramientas para el desarrollo de API REST en proyectos que utilizan Django como framework de aplicación base. Algunas de sus características incluyen:

- API explorable en la red.
- Políticas de autenticación.
- Serialización con soporte para fuentes de datos ORM²¹.
- Documentación extensiva y amplio soporte comunitario.

Django REST Framework es utilizado en el sistema GRHS para realizar operaciones sobre los datos a través de los verbos HTTP²² (*POST*, *DELETE*, *PUT*, *GET*). En la solución propuesta se utilizará para implementar un API REST en el servidor. Los servicios de dicho API proveerán acceso controlado a los datos de GRHS desde Gadmin Android.

1.7 Conclusiones del capítulo

Sobre el estudio de las soluciones informáticas que se utilizan actualmente para la realización de inventario de recursos de hardware y software se identificó que:

- Existe un predominio en la tendencia a proveer al menos un tipo de interfaz de administración compatible con dispositivos móviles. Con la excepción de Total Network

²⁰ **REST**: Representational State Transfer (Transferencia de Estado Figurativo), es un estilo de arquitectura de software para sistemas distribuidos.

²¹ **ORM**: Object-Relational Mapping (Mapeo Objeto-Relacional), es una técnica de programación para convertir datos entre sistemas de software utilizando una base de datos relacional como motor de persistencia.

²² **HTTP**: Hyper Text Transfer Protocol (Protocolo de Transferencia de Hipertexto), es el protocolo de comunicación que permite la transferencia de información en Internet.

Inventory, los sistemas analizados tratan de expandir su alcance suministrando mecanismos de usabilidad en dispositivos móviles, ya sea a través de un interfaz responsive o una aplicación nativa.

- Se verificó la validez del planteamiento inicial de la presente investigación confirmando que, el no contar con una interfaz compatible con dispositivos móviles, limita las posibilidades de explotación, y la competitividad en el ámbito comercial, de un sistema de gestión de recursos de hardware y software frente a las tendencias que predominan actualmente en el desarrollo de este tipo de aplicaciones.

Respecto al enfoque de desarrollo de aplicaciones móviles a utilizar para la implementación de la solución propuesta, se determinó que:

- Al ser el enfoque nativo el que ofrece mejores resultados en los ámbitos de rendimiento, experiencia de usuario y funcionalidad, es el más idóneo para la implementación de la solución propuesta, debido que estos aspectos son los que presentan mayores problemas actualmente en el acceso a **gadmin** desde dispositivos móviles.
- El uso de este enfoque abre las puertas a todo un conjunto de funcionalidades dirigidas a explotar las capacidades de Android y que en el futuro podrían añadirse al sistema GRHS, aumentando de esta forma su valor como producto de software.

Con la adopción del enfoque ágil y la metodología de desarrollo XP, se prevé una mayor adaptabilidad a los cambios que ocurren continuamente en los requisitos de un proyecto como el que se propone en el presente ejercicio académico.

Capítulo 2: Planificación y diseño del sistema

2.1 Introducción

En este capítulo se explican los principales aspectos de diseño de la aplicación Gadmin Android. Su contenido incluye la descripción del modelo de propuesta de solución así como las fases de **Planeación** y **Diseño** correspondientes a la metodología de desarrollo de software seleccionada.

2.2 Modelo de la propuesta de solución

La solución propuesta está diseñada para aquellas entidades que hagan uso del sistema GRHS, y dispositivos móviles que operan con sistema operativo Android, como recurso institucional y que posean a su vez una infraestructura de red adecuada. Su objetivo es permitir la consulta de la información relacionada con los inventarios de los recursos de hardware y software, así como el monitoreo de estadísticas y la gestión de incidencias, desde cualquier punto geográfico dentro del alcance de la red inalámbrica de la entidad. La solución debe permitir el control del estado de los recursos de hardware y software, así como del cumplimiento de las políticas de seguridad informática establecidas.

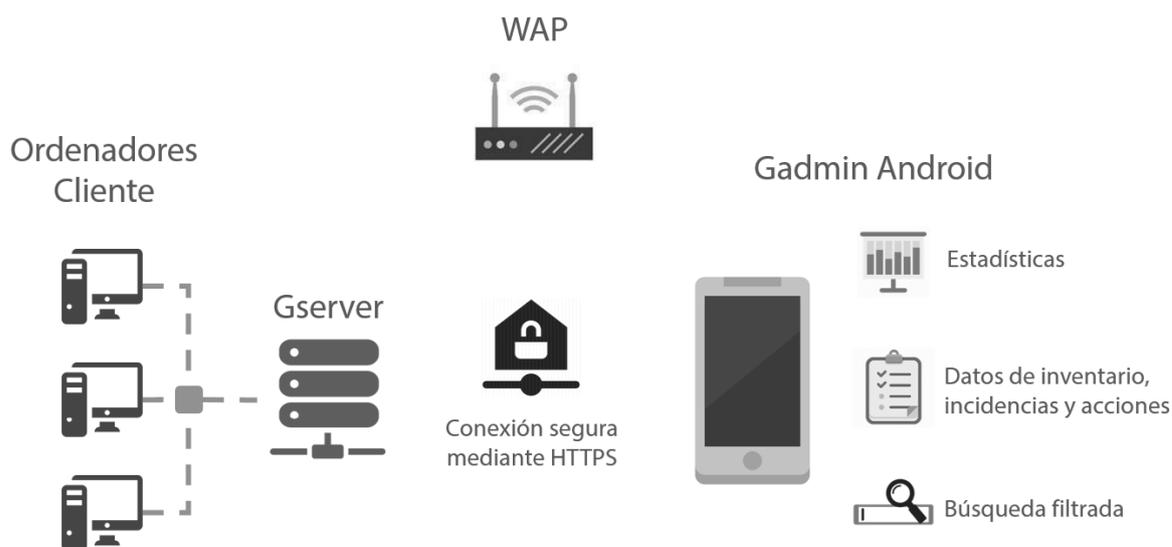


Figura 3 Modelo de la propuesta de solución.

Gadmin Android será capaz de mostrar de manera detallada los datos de cada uno de los agentes conectados a la red, que posean una instancia de **gclient** instalada y que el usuario esté autorizado a consultar; incluyendo su ubicación, dirección IP²³, estado del recurso (activo o inactivo) así como el inventario de sus componentes de hardware y software. Los datos obtenidos del servidor podrán consultarse aun cuando no haya conexión a la red de la institución, mediante un sistema de sincronización y una base de datos local en el dispositivo.

²³ **IP:** Internet Protocol (Protocolo de Internet), conjunto de reglas técnicas que controlan la comunicación en Internet.

Las solicitudes de información al servidor serán realizadas mediante el protocolo HTTPS²⁴, asegurando así el intercambio seguro de las credenciales de autenticación. En cada solicitud se incluirá el nombre del usuario que la realiza y un token²⁵ de seguridad que se almacena de forma persistente en el servidor. La aplicación permitirá la configuración de parámetros de la cuenta tales como la dirección donde se encuentra el servidor de GRHS y el dominio de autenticación al que pertenece el usuario.

2.2.1 Sincronización de datos

La sincronización de datos es el proceso de copiar automáticamente un conjunto de datos entre uno o más dispositivos (o sistemas), de tal forma que se mantenga consistente la información en uno y otro. La idea clave, detrás de la implementación de tal mecanismo, es evitar la realización de tareas de descarga de datos desde la red o de manipulación de datos persistentes dentro del hilo principal de la aplicación. El hilo principal en las aplicaciones Android es donde se ejecuta la interfaz de usuario. Como se explica en (16), las operaciones de red pueden implicar retrasos impredecibles por lo que, para evitar que esto cause una pobre experiencia de usuario, se recomienda realizarlas en un hilo de ejecución diferente.

El desarrollo de aplicaciones Android, que implementan un mecanismo de sincronización, requiere de la creación de una estructura centrada en un *Sync Adapter* (17), componente que encapsula el código para las tareas de transferencia de datos entre el dispositivo y un servidor.

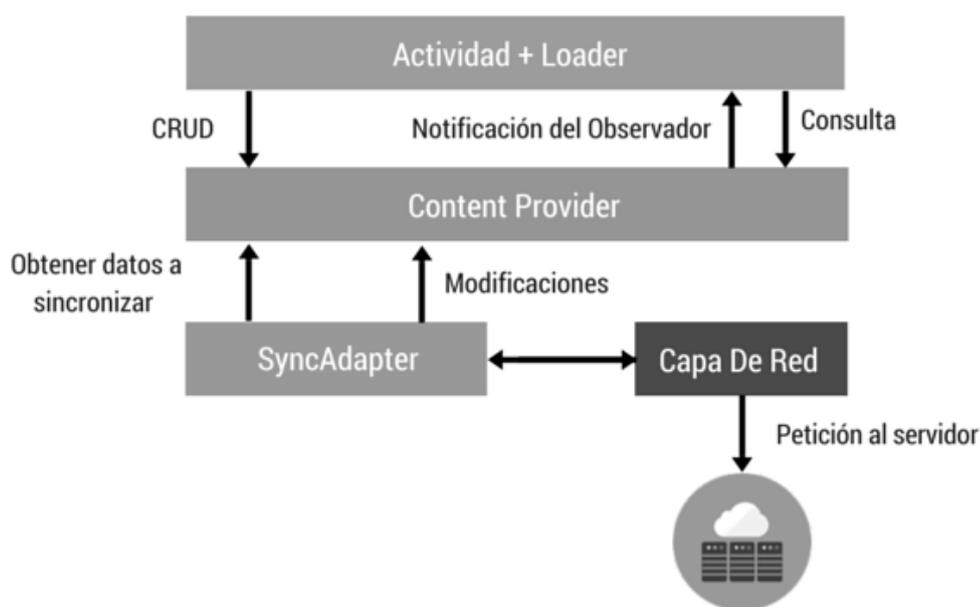


Figura 4 Estructura centrada en un Sync Adapter.

²⁴ **HTTPS:** es un protocolo de comunicación, mediante conexiones seguras, utilizado ampliamente en Internet.

²⁵ **Token:** en sistemas computacionales, es un objeto que contiene las credenciales de seguridad para una sesión de autenticación e identifica al usuario, los grupos a los que pertenece y sus privilegios.

Un *Sync Adapter* se caracteriza por realizar las acciones asincrónicamente, es decir, en periodos de tiempo sin inicio o fin determinados, por lo que es posible que la transferencia de datos no suceda cuando se espera; sin embargo, la integridad de la información se mantiene. Dicho mecanismo se utilizará en Gadmin Android para las operaciones de transferencia de datos entre Gadmin Android y **gserver**.

2.2.2 Autenticación

Android incluye un componente llamado *Authenticator*, para manejar las cuentas asociadas cada aplicación, el cual es vital para el esquema de sincronización (18). El framework de sincronización asume que la transferencia ocurre entre datos almacenados en el dispositivo y datos almacenados en un servidor remoto que requiere autenticación, por esa razón requiere que se incluya el componente *Authenticator* como parte de la estructura del *Sync Adapter*. Entre los beneficios asociados su implementación se encuentran:

- Simplificación de la autenticación para el desarrollador
- Forma estándar de autenticación de usuarios
- Seguridad en el almacenamiento y uso de credenciales
- Manejo de casos de acceso denegado

En la solución propuesta se implementará este componente para gestionar la creación de la cuenta local asociada a la aplicación y el manejo seguro de las credenciales de autenticación de un usuario del sistema GRHS. Estas credenciales se utilizarán para la obtención del token de seguridad necesario para la realización de solicitudes al servidor.

2.2.3 Consulta y manipulación de datos

La consulta y manipulación de los datos almacenados en la base de datos local de Gadmin Android se realizará mediante un componente llamado *Content Provider*, el cual, según se explica en (19), gestiona el acceso a un repositorio central de datos y ofrece una interfaz estándar y consistente para los mismos. Este componente maneja la comunicación entre procesos y el acceso seguro a los datos.

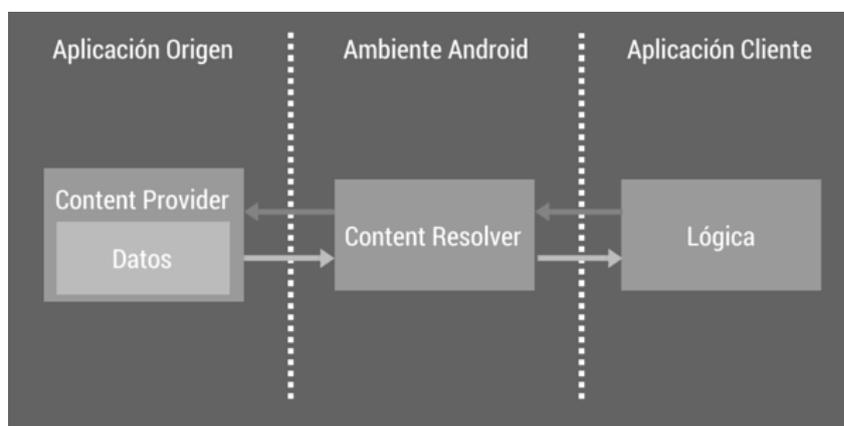


Figura 5 Estructura lógica de un Content Provider.

La aplicación accede a los datos de un *Content Provider* mediante un objeto cliente *Content Resolver*. Dicho objeto provee los métodos CRUD²⁶ básicos para el almacenamiento persistente y accede a los datos del *Content Provider* mediante la revisión de las URI²⁷ de contenido que el segundo utiliza para encapsular su implementación SQLite²⁸, protegiendo así los datos del acceso no autorizado por parte de aplicaciones externas.

2.2.4 Gestión de almacenamiento de datos en el dispositivo

Tomando en consideración el gran tamaño que la base de datos de GRHS puede llegar a alcanzar (con 4000 clientes registrados, en su versión de pruebas, llega a ocupar alrededor de 2 GB²⁹) y que usualmente los dispositivos móviles cuentan con un espacio de almacenamiento interno más limitado que el de los ordenadores tradicionales, se hace vigente la necesidad de implementar un mecanismo que permita controlar cuánta información puede descargar y mantener almacenada la aplicación Gadmin Android.

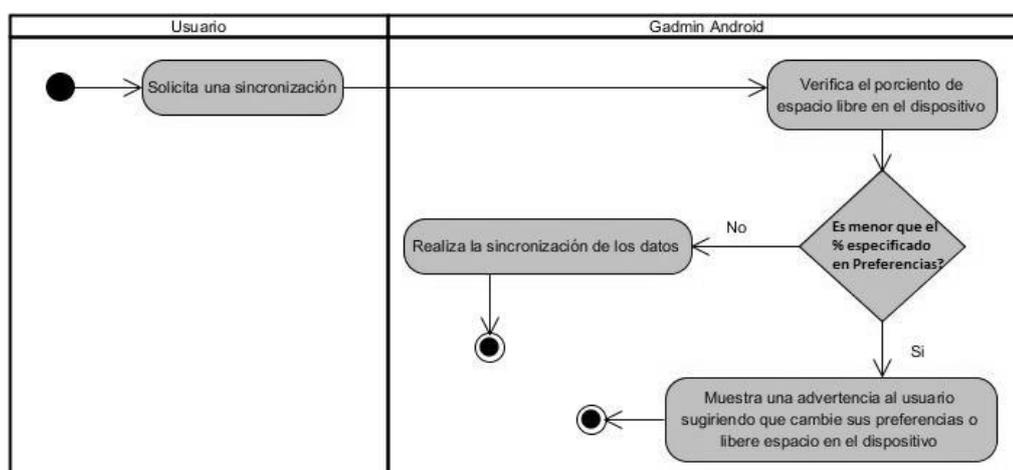


Figura 6 Secuencia de verificación de espacio disponible en el dispositivo móvil durante una sincronización.

Ante cada solicitud de sincronización realizada por el usuario, el sistema primero verificará un parámetro en las preferencias de la aplicación. Dicho parámetro corresponde al porcentaje de espacio libre mínimo que debe existir en el dispositivo para autorizar una descarga de datos del servidor. Si el porcentaje de espacio libre, en el momento de la solicitud, es menor que este parámetro, la aplicación abortará el proceso de sincronización y mostrará al usuario un mensaje

²⁶ **CRUD**: Create, Read, Update, Delete (Crear, Leer, Actualizar, Eliminar), es un acrónimo que, en ciencias computacionales, se utiliza para referirse a las funciones básicas que implementa la capa de persistencia de datos de un software determinado.

²⁷ **URI**: Uniform Resource Identifier (Identificador de Recursos Uniforme), es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

²⁸ **SQLite**: sistema de gestión de bases de datos relacionales comúnmente utilizado por aplicaciones Android para implementar su capa de persistencia de datos.

²⁹ **GB**: gigabyte, es un múltiplo de la unidad de medida de almacenamiento de información digital, byte, que equivale a 10⁹ bytes.

advirtiéndole que cambie sus preferencias o libere espacio en su dispositivo. De esta forma se evita la ocurrencia de errores de Lectura/Escritura y se mantiene la integridad de los datos.

2.3 Planeación

En la etapa de **Planeación** comienza la interacción con el cliente para identificar los requerimientos del sistema y se identifican las iteraciones y ajustes necesarios a la metodología según las características de la solución. En la misma se describen tres elementos fundamentales, los cuales son: Historia de Usuario (HU), Tiempo de Ejecución del Proyecto y Plan de Entrega, en ellos se recogen las características principales del sistema a construir y se concreta el tiempo que se necesitará para implementarlas.

2.3.1 Historias de usuario

Kent Beck (20) define a las HU como una parte importante de XP. Constituyen el cimiento de los requisitos del sistema; son el artefacto primario utilizado para definir lo que el equipo de desarrollo estará implementando en una iteración dada. El cliente es el encargado de asignar una prioridad a cada HU y es el equipo de desarrollo el encargado de asignar un costo, que se traduce en las semanas que llevará el desarrollo de las mismas. En la tabla siguiente se muestra un ejemplo de las HU especificadas para el desarrollo de Gadmin Android. Las HU restantes se encuentran en el **Anexo II Historias de Usuario**.

Tabla 2 HU-1 Implementar un API REST en el servidor de GRHS

Historia de usuario	
Número: HU-1	Nombre de la historia de usuario: Implementar un API REST en el servidor de GRHS
Modificación de la historia de usuario: 3	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración asignada: 1
Prioridad en el negocio: Alta	Puntos estimados: 2
Descripción: El API se compone de un conjunto de servicios que devuelven los datos de GRHS ante solicitudes desde aplicaciones externas al servidor. Cuando un usuario de GRHS desee consumir alguno de los servicios, debe primero realizar una solicitud POST al servicio de autenticación con su nombre de usuario, contraseña y dominio de autenticación como parámetros. Si el usuario y sus credenciales son correctas, el servidor devuelve el token de seguridad correspondiente. Entonces el usuario puede realizar una nueva solicitud, esta vez tipo GET, enviando su nombre de usuario y token de seguridad como parámetros, al servicio del que desea obtener los datos, los cuales se dan en formato JSON.	
Observaciones: Ninguna	
Prototipo de interfaz: No procede	

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.

Leyenda de las Historias de Usuario:

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: 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, donde un punto de estimación corresponde a una semana de 5 días con 8 horas de trabajo diarias.

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 historia de usuario.

De acuerdo a lo planteado por Cohn y Paul (21), en XP no es necesario desarrollar una especificación formal de requisitos del sistema. En su lugar, se espera que un conjunto de HU informales realice esa función. Las HU no son un conjunto único, normalizado, vinculado, completo y consistente. Sin embargo, satisfacen las propiedades de ser abstractas, limitadas, modificables, trazables, comprobables, configurables y granulares. El concepto de HU de XP también satisface la mayor parte de los requisitos de categorización (Identificación, Prioridad, Criticidad, Viabilidad, Riesgo y Fuente) de la guía IEEE³⁰.

2.3.2 Características no funcionales del sistema

Aun cuando XP confía en las HU como principal artefacto para la especificación de los requisitos del sistema, Eberlein y Leite plantean en (22) que, si se consideran los aspectos no funcionales sólo

³⁰ **IEEE:** Instituto de Ingeniería Eléctrica y Electrónica, asociación mundial de ingenieros dedicada a la estandarización y el desarrollo en áreas técnicas.

en el nivel de implementación, como se entiende que es el caso de las metodologías ágiles, entonces los problemas serán aún mayores que cuando se tienen en cuenta sólo en el nivel de diseño. Para evitar esto, se realiza una especificación formal de las características no funcionales de la solución propuesta.

Conexión segura: la conexión entre Gadmin Android y los servicios del API REST de GRHS debe realizarse mediante el protocolo HTTPS, en pos de proteger las credenciales de autenticación que se incluyen en la cabecera HTTP de cada solicitud.

Consistencia de los datos: los datos mostrados en Gadmin Android mantendrán la consistencia con los datos almacenados en la base de datos de GRHS, esto quiere decir que al descargar un objeto desde el servidor (por ejemplo, una incidencia), la aplicación verificará si el mismo ya existe en la base de datos local y lo actualizará si ha sufrido algún cambio. Esto evitará la existencia de datos duplicados y la realización de inserciones innecesarias.

Versión de Android: el dispositivo donde se instalará la aplicación debe tener instalado una versión de Android igual o posterior a la versión 4.0. Esta es la versión mínima con soporte completo en el SDK utilizado por el equipo de desarrollo.

Cuenta única: solo existirá una cuenta de GRHS a la vez en un mismo dispositivo. Al borrar una cuenta, se eliminará todos los datos que se hayan descargado desde la misma.

Soporte multilingaje: en su primera versión, la aplicación Gadmin Android incluirá los lenguajes Inglés y Español, con el primero como lenguaje por defecto.

2.3.3 Tiempo de ejecución del proyecto

El tiempo de ejecución del proyecto es un elemento que le permite al equipo de desarrollo completar las historias de usuario en una determinada iteración. Este elemento se calcula sumando los puntos estimados de las historias de usuario realizadas en una iteración. En este sentido un punto estimado equivale a una semana ideal de programación donde se trabaje sin interrupciones, ocho horas diarias durante cinco días. Dado que la planificación se realiza basándose en el tiempo de implementación de una historia de usuario, el tiempo de ejecución se utiliza para declarar cuántas historias de usuario se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de las mismas. La tabla siguiente muestra la estimación del costo en tiempo realizada a partir de la especificación de las HU, la cual provee una idea de lo que tardará el desarrollo de la solución propuesta en el presente trabajo.

Tabla 3 Estimación del costo en tiempo por historia de usuario

Numero	Nombre de la historia de usuario	Puntos estimados
HU-1	Implementar un API-REST en el servidor de GRHS	2
HU-2	Implementar un mecanismo de manejo de permiso para el acceso al API-REST	1
HU-3	Implementar un mecanismo de gestión de cuenta de usuario	1
HU-4	Implementar un mecanismo de descarga y sincronización de datos a demanda	2
HU-5	Mostrar estadísticas del sistema	1
HU-6	Mostrar información de agentes, inventario, acciones, incidencias	1
HU-7	Implementar una interfaz para el manejo de preferencias de la aplicación	1
HU-8	Gestionar la conexión de la aplicación mediante el protocolo HTTPS	1
HU-9	Implementar un mecanismo de búsqueda filtrada	1

La estimación realizada da como resultado una duración aproximada de tres meses para el desarrollo de la solución propuesta. Este resultado permite dividir el proceso de desarrollo en etapas para facilitar su realización. Letelier y Penadés (10) describen estas etapas en la tercera fase de XP y las denominan iteraciones.

2.3.4 Iteraciones

Para cada iteración, se define un número determinado de historias de usuario a implementar y al final de cada iteración se obtiene como resultado la entrega de las funcionalidades correspondientes. La duración de cada iteración no debe ser superior a tres semanas. Esto ocurre conjuntamente con la aceptación, por parte del cliente, de los requerimientos solucionados. Idealmente, al final de la última iteración el sistema estará listo para entrar en producción. La tabla siguiente muestra las iteraciones en que estará dividido el proceso de desarrollo de la solución propuesta en el presente trabajo, así como la distribución de las HU a desarrollar por cada iteración definida.

Tabla 4 Plan estimado de duración de las iteraciones

Iteración	Título de Historia de Usuario	Duración total
1	<ol style="list-style-type: none"> 1. Implementar un API-REST en el servidor de GRHS 2. Implementar un mecanismo de manejo de permiso para el acceso al API-REST 	3
2	<ol style="list-style-type: none"> 3. Implementar un mecanismo de gestión de cuenta de usuario 4. Implementar un mecanismo de descarga y sincronización de datos a demanda 	3
3	<ol style="list-style-type: none"> 5. Mostrar estadísticas del sistema 6. Mostrar información de agentes, inventario, acciones, incidencias 7. Implementar una interfaz para el manejo de preferencias de la aplicación 	3
4	<ol style="list-style-type: none"> 8. Gestionar la conexión de la aplicación mediante el protocolo HTTPS 9. Implementar un mecanismo de búsqueda 	2

Los resultados anteriores permiten estimar la duración de cada iteración, utilizando para ello los puntos estimados de las HU correspondientes. Esto permite elaborar un plan de entrega de funcionalidades implementadas al final de cada iteración.

2.3.5 Plan de entrega

El plan de entrega consiste en establecer, de manera conjunta entre el equipo de desarrollo y el cliente, la duración y fecha de entrega de cada iteración hasta lograr el producto final. En este plan se detalla la fecha fin de cada iteración, mostrando una versión desarrollada del producto en ese momento, hasta lograr el producto final en la fecha establecida. En la siguiente tabla se desglosa el plan de entrega para las funcionalidades implementadas en cada iteración del proceso de desarrollo de Gadmin Android.

Tabla 5 Plan de entrega por iteraciones

Iteraciones	Fecha de entrega
1	8 abril
2	29 de abril
3	20 de mayo
4	3 de junio

2.4 Diseño de la propuesta de solución

En esta sección se realiza la descripción de la arquitectura de software, los patrones de diseño aplicados a la propuesta de solución y la descripción de las principales clases de la aplicación. Estos elementos muestran una vista técnica y lógica del software a desarrollar, describen la arquitectura utilizada, basada en las necesidades de la solución, así como las clases y sus relaciones; además, son el punto de partida para la posterior codificación.

2.4.1 Arquitectura de software

La arquitectura de software de un sistema o programa de computación es la estructura o estructuras del sistema, que comprenden elementos de software, las propiedades externamente visibles de estos elementos, y las relaciones entre ellos. (23)

Arquitectura Cliente/Servidor

Según Microsoft (24), el estilo arquitectónico Cliente/Servidor describe sistemas distribuidos que implican un sistema independiente de clientes y servidores, y una red que los conecta. La forma más simple de sistema Cliente/Servidor consiste en una aplicación servidor a la que se accede directamente por varios clientes, a la cual se refiere como un estilo arquitectónico de 2 niveles.

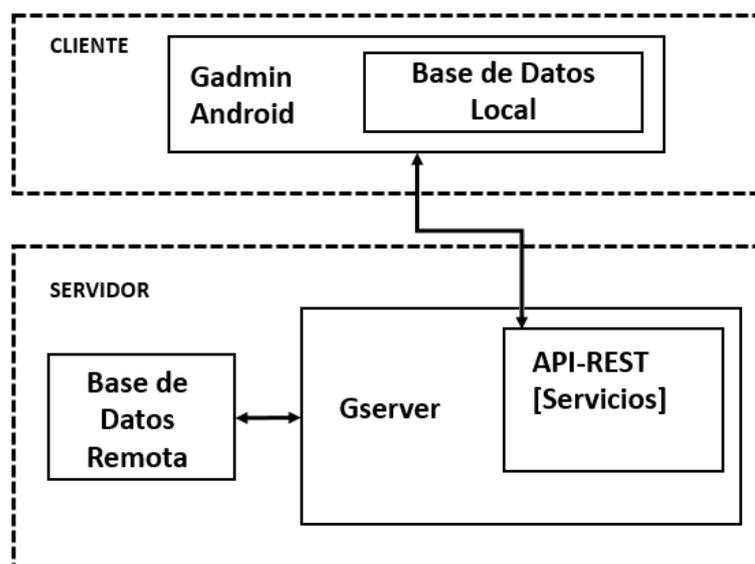


Figura 7 Arquitectura Cliente/Servidor desglosada en sus dos niveles.

En el contexto de la solución propuesta, la aplicación Gadmin Android ocupa el nivel Cliente, ya que la misma será el demandante de información del servidor de GRHS y es el software que se ejecuta del lado del usuario. El modelo de solución plantea un escenario en el que múltiples instancias de Gadmin Android solicitan información a **gserver**, aplicación principal del nivel Servidor, y este se encarga de decidir qué información proveer según los privilegios del usuario que hace la solicitud.

2.4.2 Patrones de diseño

Un patrón de diseño provee un esquema para refinar componentes de un sistema de software y la forma en que se relacionan entre sí.

De manera más simple, un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos. (25)

Puede asumirse que un patrón describe una estructura de comunicación de componentes, generalmente recurrente, que resuelve un problema de diseño general dentro de un contexto particular. Por tanto un patrón de diseño es una solución demostrada, eficiente y reutilizable en diferentes contextos y escenarios de desarrollo de software.

Patrones GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns) son un conjunto de patrones de diseño de software que:

Describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. (25)

Los patrones pertenecientes a este conjunto son: Experto, Creador, Alta Cohesión y Bajo Acoplamiento. En la siguiente sección se explica el uso de algunos de estos patrones en el diseño de Gadmin Android.

Experto

El patrón Experto en Información establece el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto, o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. El comportamiento se distribuye entre las clases que cuentan con la información requerida (25). La siguiente figura muestra un ejemplo de este patrón en la relación entre las clases AccountAuthenticator y AuthenticationActivity.

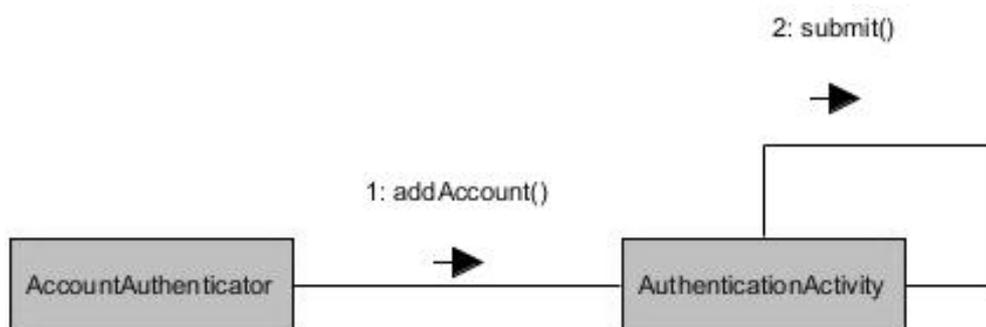


Figura 8 Solicitud de creación de una nueva cuenta.

La clase `AuthenticationActivity` contiene los atributos y métodos específicos para el manejo de la cuenta de usuario de GRHS, lo cual le permite crear o eliminar cuentas de usuario en el dispositivo donde se ejecuta la aplicación, guardar sus credenciales de manera segura y cambiar los datos de una cuenta existente. La misma contiene mecanismos para atender las solicitudes de creación, cambio y eliminación de las cuentas de usuario de la aplicación, realizadas por clases como `AccountAuthenticator`. Dado que la clase `AuthenticationActivity` es la que contiene toda la información necesaria para la creación de una nueva cuenta, es en ella donde se implementa el método para crearla, por tanto es el Experto en Información para este caso en particular.

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. (25)

La siguiente muestra un ejemplo de la aplicación de este patrón en la clase `SyncService`, perteneciente al paquete `cu.uci.gadmin.sync`.

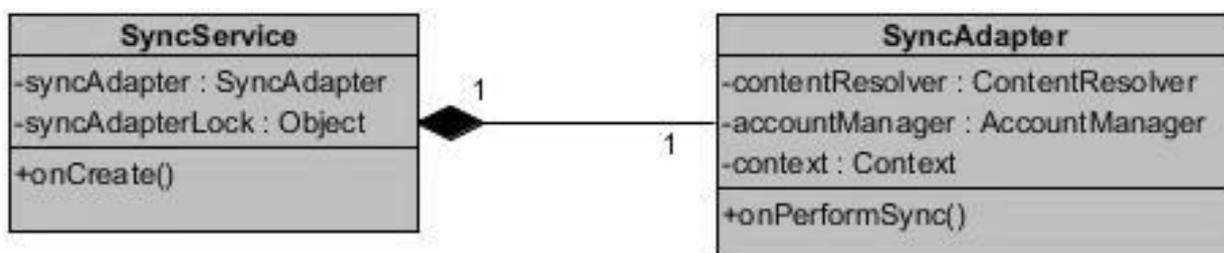


Figura 9 Relación de composición entre las clases `SyncAdapter` y `SyncService`.

La clase `SyncService` es la única responsable de la creación de objetos de la clase `SyncAdapter`. Entre ambas clases se establece una relación de composición con cardinalidad uno a uno.

Alta Cohesión

La alta cohesión indica que los datos y responsabilidades de una entidad están fuertemente ligados a la misma en un sentido lógico. 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. Este patrón permite la obtención de clases fácilmente actualizables y reutilizables durante el proceso de desarrollo de software. Según se plantea en (25), se considera que un diseño de clases provee una alta cohesión cuando:

Una clase tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas. (25)

Un diseño con alta cohesión puede observarse en la relación de las clases del paquete **cu.uci.gadmin.providers**, donde se maneja la capa de acceso a datos locales en Gadmin Android. La clase `DataBaseContract` contiene la definición estructural de la base de datos local de la aplicación Gadmin Android, así como las URI de contenido y los tipos MIME de cada dato que se maneja en la misma. Colabora con las clases `DataBaseHelper`, responsable de la creación y actualización del esquema de la base de datos, y `DataBaseContentProvider`, clase que implementa los métodos necesarios para realizar las operaciones sobre los datos locales.

Patrones Gang Of Four o Pandilla de los Cuatro

Los patrones Gang Of Four (GOF) son un conjunto de patrones de diseño muy útiles durante el diseño de objetos. Estos se clasifican según el propósito para el que han sido definidos como: Patrones creacionales, estructurales y de comportamiento.

Builder

Para utilizar un *Content Provider* en la realización de múltiples operaciones consecutivas sobre la base de datos local de Gadmin Android, la práctica recomendada en (19) es realizarlas en un lote de operaciones. Para ello se recomienda el uso de la clase `ContentProviderOperations` como mecanismo para construir las acciones sobre los datos de la aplicación tales como: insertar, eliminar y actualizar. En la figura siguiente se ilustra un esquema de este mecanismo.

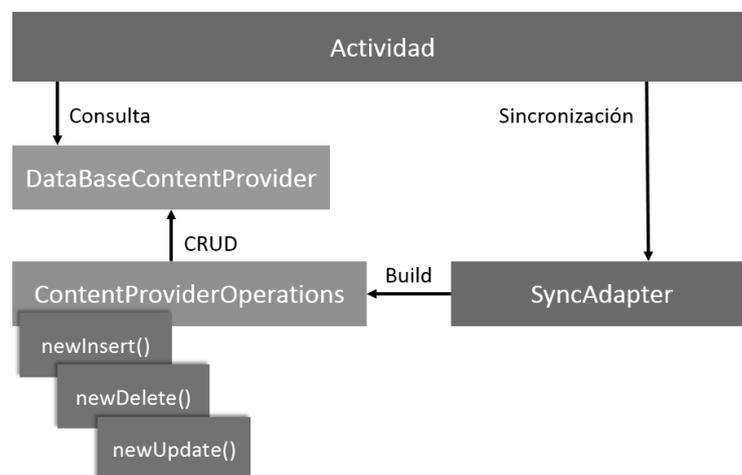


Figura 10 Uso de la clase `ContentProviderOperations` para realizar operaciones sobre la capa de acceso a datos persistentes de Gadmin Android.

Para crear un objeto de la clase `ContentProviderOperation` es necesario construirlo usando la clase interna `Builder`. Esta clase es un ejemplo del patrón GOF Builder, el cual es un patrón creacional que separa el proceso de construcción de un objeto complejo de su representación, para que el mismo proceso de construcción pueda crear diferentes representaciones.

Patrón Maestro/Esclavo

El patrón de diseño Maestro/Esclavo es usado cuando se tienen dos o más procesos que necesitan ejecutarse simultánea y continuamente pero a diferentes velocidades. National Instruments (26) lo describe como un patrón de diseño que consiste en múltiples ciclos paralelos. Cada ciclo puede ejecutar tareas a velocidades distintas. De estos, uno actúa como el maestro y los otros como esclavos. El ciclo maestro controla todos los ciclos esclavos y se comunica con ellos utilizando arquitecturas de mensajería.

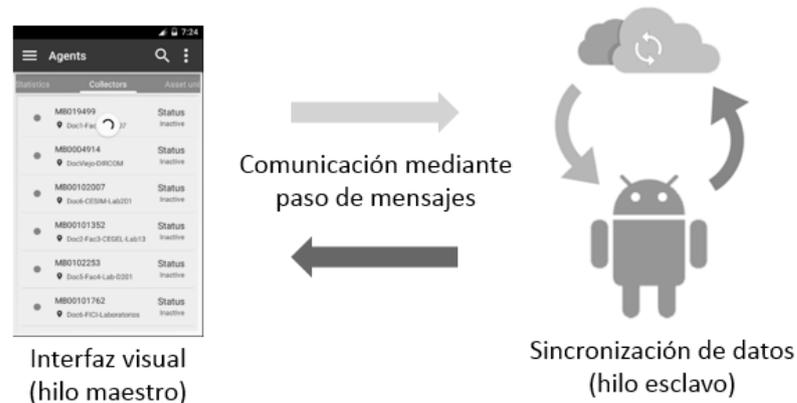


Figura 11 Esquema de aplicación del patrón Maestro/Esclavo en Gadmin Android.

En el diseño de Gadmin Android, existen dos hilos de ejecución que se ejecutan simultáneamente en el proceso principal de la aplicación. Uno de estos hilos es el de la interfaz visual, que maneja toda la lógica de presentación de los datos al usuario. El segundo hilo es el de sincronización de los datos, que se encarga de la lógica de descarga de datos del servidor, y la manipulación de los datos locales. El hilo maestro es el de interfaz visual, ya que este es el que controla la ejecución del hilo de sincronización mediante las acciones del usuario. La comunicación entre ambos hilos se establece mediante el uso de Intents.

2.4.3 Tarjetas Clase-Responsabilidad-Colaborador

Scott Ambler (27) define las tarjetas Clase-Responsabilidad-Colaborador (CRC) como tarjetas de índice estándar que han sido divididas en tres secciones, una que indica el nombre de la clase que la tarjeta representa, un inventario de las responsabilidades de la clase, y la tercera con los nombres de las otras clases con las que ésta colabora para cumplir su responsabilidades.

En el Manifiesto para el Desarrollo Ágil de Software (28) se recomienda el uso de este artefacto para inventariar las clases que son relevantes para el incremento actual del software. La siguiente tabla muestra una de las tarjetas CRC elaboradas para el modelo de la solución propuesta en el presente trabajo. Las tarjetas CRC restantes se encuentran en el **Anexo II Tarjetas CRC**.

Tabla 6. Tarjeta CRC AccountAuthenticator

Nombre de la clase: AccountAuthenticator
Responsabilidades de la clase: Se encarga de manejar las operaciones relacionadas a las cuentas de usuario de la aplicación. Esta clase sabe qué actividad mostrar para que el usuario ingrese sus credenciales y dónde encontrar algún token retornado por el servidor previamente.
Colaboradores de la clase: <ul style="list-style-type: none"> • AuthenticationActivity • CustomSSLSocketFactory • CustomX509TrustManager • Constants

Conclusiones del capítulo

En este capítulo se elaboraron los artefactos correspondientes a la etapa de planeación de la metodología XP, determinando como resultado que el proceso de desarrollo durará un total de 11 semanas, divididas en cuatro iteraciones de 3, 3, 3 y 2 semanas respectivamente, de forma 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.

Se especificó la arquitectura de software y los patrones de diseño, lo que permitió el diseño de una aplicación que podrá ser mantenida, auditada y actualizada con relativa facilidad, por parte de los desarrolladores en futuras versiones. Se confeccionaron además las tarjetas CRC, las cuales permitieron establecer el marco de relaciones para las clases del diseño. Estos artefactos ayudarán a una mejor comprensión de la manera en que se encuentra estructurada la aplicación, lo cual facilita su análisis y, de ser necesario, su modificación.

Capítulo 3: Implementación y pruebas del sistema

3.1 Introducción

En el presente capítulo se detallan los aspectos fundamentales de los procesos de implementación y pruebas de la solución propuesta en el Capítulo 2. Se describen los estándares de codificación utilizados durante la implementación de la aplicación Gadmin Android y del API REST de GRHS, el diagrama de despliegue de la solución, así como los artefactos generados durante la etapa de pruebas.

3.2 Estándares de codificación

Los estándares de codificación establecen las pautas de programación enfocadas en la estructura y apariencia física del código para facilitar la legibilidad, comprensión y mantenimiento del mismo. Kent Beck (20) recomienda el uso de estándares de codificación, como una práctica de XP, en la que los programadores escriben todo el código de conformidad con normas que enfatizan la comunicación a través del código. Dichos estándares definen la nomenclatura de las variables, objetos, métodos y funciones.

Java CamelCase

La notación *CamelCase* es la práctica de escribir palabras compuestas, o frases, tales que cada palabra o abreviatura comienza con una letra mayúscula y omite guiones. *CamelCase* puede comenzar con una letra mayúscula, estilo que es también conocido como *UpperCamelCase* y que es usado sobre todo para nombrar clases, o con una letra minúscula para nombrar variables y métodos. Es un estilo recomendado por varias comunidades y empresas desarrolladoras de Java, tales como Oracle Corporation (29); además, este estilo de codificación está integrado directamente en el sistema de auto completamiento de código de Android Studio, lo cual acelera el proceso de escritura del código manteniendo su coherencia y legibilidad.

PEP 8

PEP 8 es el nombre dado a la Guía de Estilo para Código Python (30) (31), creada por Guido van Rossum, Barry Warsaw y Nick Coghlan. Una de las ideas clave de Guido es que el código es leído con mucha más frecuencia de lo que es escrito. Las directrices que provee están destinadas a mejorar la legibilidad del código y hacer que sea coherente en todo el amplio espectro de código Python. Esta guía es la que regirá el código escrito en la implementación del API REST de GRHS, ya que al estar incluido dentro del código fuente de **gserver**, que es un proyecto escrito en Python, contribuye a mantener la consistencia en el mismo.

3.2.1 Nomenclatura de las clases

Los nombres de las clases, tanto en el código de Gadmin Android como en el del API REST, comienzan con la primera letra en mayúscula. Si el nombre es compuesto, la primera letra de cada

una de las palabras subsiguientes se escribe en mayúsculas, como se muestra en las siguientes figuras.

```
48
49 class GetHostData(generics.ListAPIView):
50     """
51     View that returns all possible information about all collectors,
52     including hardware and software inventory active data
53     """
```

Figura 12 Declaración de una clase en el código fuente del API REST de GRHS.

```
65 /**
66  * Created by nacastruc on 4/29/16.
67  * Manages Client - Server data transfer and synchronization
68  */
69 public class SyncAdapter extends AbstractThreadedSyncAdapter {
```

Figura 13 Declaración de una clase en el código fuente de Gadmin Android.

Esta convención de nomenclatura de las clases cumple tanto con las pautas *CamelCase* como con las de *PEP 8*, lo cual permite usarlas en ambos productos de software sin violar las convenciones descritas anteriormente.

3.2.2 Nomenclatura de los métodos y variables en el código de Gadmin Android

Los nombres de los métodos y variables en el código de Gadmin Android, siguiendo las pautas de *CamelCase*, comienzan con la primera letra en minúscula, si el nombre es compuesto la primera letra de cada una de las palabras subsiguientes se escribe en mayúsculas, como se muestra en la siguiente figura.

```
98 private static Account getAccountToSync(Context context) {
99     // Get the AccountManager
100     AccountManager accountManager = (AccountManager) context
101         .getSystemService(Context.ACCOUNT_SERVICE);
102     // Get accounts of specified type
103     Account [] accounts = accountManager.getAccountsByType(Constants.ACCOUNT_TYPE);
```

Figura 14 Declaración de métodos y variables en el código fuente de Gadmin Android.

3.2.3 Nomenclatura de las constantes

Los nombres de las constantes se escriben con letras en mayúsculas, si el nombre es compuesto, se separa cada palabra utilizando guiones bajos, como se muestra en la figura.

```
7 /**
8  * Authentication
9  */
10 // Types
11 public static final String ACCOUNT_TYPE = "cu.uci.gadmin";
12 public static final String AUTH_TOKEN_TYPE = "cu.uci.gadmin.AUTH_TOKEN_TYPE";
```

Figura 15 Declaración de una constante en el código fuente de Gadmin Android.

La convención de nomenclatura descrita anteriormente se conoce comúnmente como *UPPER_CASE_WITH_UNDERSCORES*, y es utilizada en la mayoría de los lenguajes de alto nivel basados en C, como Java y Python, para el nombramiento de constantes.

3.2.4 Nomenclatura de los métodos y variables en el código del API REST de GRHS

Los métodos y las variables declaradas en el código fuente del API REST se nombrarán usando el estilo de nomenclatura *lower_case_with_underscores*, recomendado en *PEP 8*. El mismo consiste en nombrar métodos y variables con letras minúsculas. En caso de que los nombres sean compuestos, las palabras se separan con guiones bajos como se muestra en la siguiente figura:

```
56 | paginate_by = 10
57 |
58 | def get_queryset(self):
59 |     # Agent
60 |     host = self.request.QUERY_PARAMS.get('host', None)
61 |     ip_address = self.request.QUERY_PARAMS.get('ip', None)
62 |     last_ping = self.request.QUERY_PARAMS.get('last_ping', None)
63 |
```

Figura 16 Declaración de métodos y variables en el código fuente del API REST de GRHS.

El uso de esta notación permite mantener el estilo utilizado hasta ahora en el desarrollo del servidor de GRHS, contribuyendo a la legibilidad del código y facilitando su auditoría y actualización.

3.3 Diagrama de despliegue

Un diagrama de despliegue muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución en las instancias de los nodos de proceso (25). Los artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo.

El destino de despliegue está generalmente representado por un nodo que es, o bien de los dispositivos de hardware o bien de algún entorno de ejecución de software. Los nodos pueden ser conectados a través de vías de comunicación para crear sistemas en red de complejidad arbitraria.

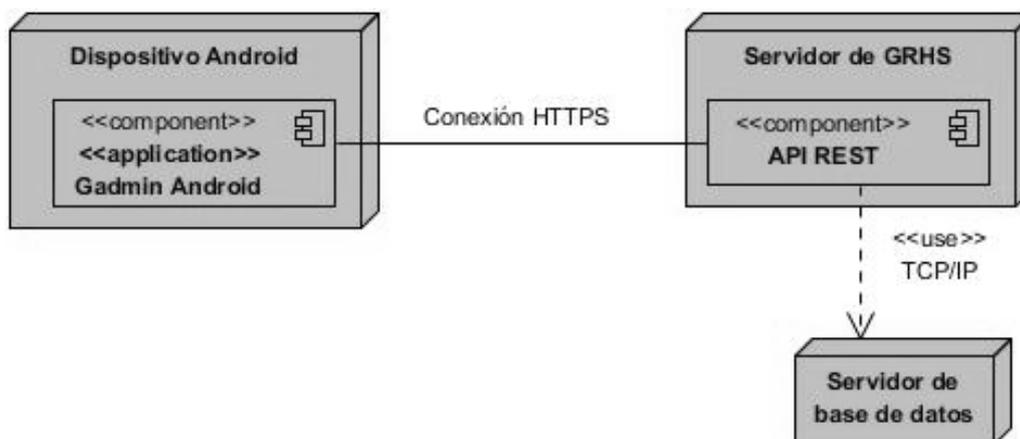


Figura 17. Diagrama de despliegue para la propuesta de solución.

Descripción de los nodos del diagrama de despliegue:

Estación Cliente: Dispositivo Android, que debe tener instalada la versión 4.0 o superior del sistema operativo y 353 MB³¹ de memoria RAM³² o más. En este nodo es donde estará la aplicación, su función es la ejecución de la aplicación y la interacción con el sistema según las necesidades del usuario.

Estación Servidor: Corresponde al servidor de aplicación, **gserver**, que contiene el API REST con los servicios que se utilizarán para acceder a los datos de los inventarios, incidencias, acciones y estadísticas del sistema.

Estación Servidor de base de datos: Representa el servidor de base de datos del sistema GRHS.

3.4 Tareas de ingeniería

Las tareas de ingenierías son definidas por Kent Beck (20) como: *algo que el programador sabe que el sistema debe hacer*. Las mismas deben ser estimables dentro de uno a tres días de programación ideales y se derivarán, en su mayoría, de historias de usuario. La siguiente tabla representa la especificación una tarea de ingeniería definida en la primera iteración del desarrollo de la solución propuesta:

Tabla 7 TI Implementar el servicio GetHostData

Tareas de Ingeniería		
Número: 1	H.U: Implementar un API REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetHostData		
Tipo: Desarrollo	Puntos estimados: 2/5	
Fecha inicio: 21 de marzo, 2016	Fecha fin: 22 de marzo, 2016	
Responsable: Nestor Antonio Castro Castillo		
Descripción: Se implementará un servicio que devuelva los datos de los colectores registrados en el servidor, así como los datos del inventario activo relacionado con cada colector (información de todos los componentes de hardware y software relacionados con el mismo).		

Letelier y Penadés (10) plantean que las tareas de ingeniería expresan todo el trabajo realizado durante una iteración. Aunque cada una de ellas son asignadas a un solo programador, son llevadas a cabo por parejas de programadores. El resto de las tareas de ingeniería definidas para cada

³¹ **MB:** megabyte, es un múltiplo de la unidad de medida de almacenamiento de información digital, byte, que equivale a 10⁶ bytes.

³² **RAM:** Random Access Memory (Memoria de Acceso Aleatorio), es una forma volátil de almacenamiento de datos digitales que permite a un dispositivo acceder a los datos (leer o escribir) en casi la misma cantidad de tiempo, independientemente de la ubicación física de los datos dentro de la memoria.

iteración del proceso de desarrollo de la solución propuesta se encuentran en el **Anexo III Tareas de ingeniería**.

3.6 Pruebas unitarias

La producción de código en XP está dirigida por la realización de pruebas unitarias, que son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema (10). Las pruebas unitarias entran en la categoría de pruebas de caja blanca, ya que son realizadas directamente sobre el código fuente de la aplicación y se realizan con el objetivo fundamental de probar la funcionalidad del código.

Durante el desarrollo de la presente solución se emplearon las herramientas PyUnit, para la implementación de los casos de prueba del código Python escrito en el API REST de GRHS, y JUnit, para la implementación de los casos de prueba del código Java escrito en Gadmin Android. Las mismas permitieron realizar pruebas a partes específicas del código mediante clases especiales. Dichas clases se conocen como *test cases* y deben ser implementadas como subclases de la clase *TestCase*, o alguna subclase de esta, en el framework de pruebas correspondiente. En la siguiente figura se muestra una de las clases de este tipo implementadas para la realización de pruebas unitarias al API REST de GRHS.

```
14 class AuthenticationTest(unittest.TestCase):
15     def setUp(self):
16         # A Client instance to perform the test
17         self.client = Client()
18
19     def test_authentication(self):
20         # Issue a POST request with correct credentials
21         response = self.client.post('/api/api-token-auth/',
22                                     {'username': 'grhs', 'password': 'grhs', 'domain': 'local'})
23         # Check that the response is 200 OK.
24         self.assertEqual(response.status_code, 200)
25
26         # Issue a post request with incorrect credentials
27         response = self.client.post('/api/api-token-auth/',
28                                     {'username': 'nacastro', 'password': '', 'domain': 'uci.cu'})
29         # Check that the response is not 200 OK.
30         self.assertNotEqual(response.status_code, 200)
31
```

Figura 18 Clase de pruebas para comprobar el funcionamiento del servicio de autenticación del API REST de GRHS mediante el framework PyUnit.

Las clases de pruebas permiten comprobar la funcionalidad de un método en casos de prueba que contemplan varios escenarios. En la clase *AuthenticationTest*, por ejemplo, se pone a prueba el servicio de autenticación, verificando que la respuesta a la solicitud con credenciales correctas devuelve el código de respuesta HTTP 200, que indica que la solicitud se completó exitosamente y se retornó el token de seguridad. Un escenario alternativo contempla el envío de una solicitud con una contraseña incorrecta y se verifica que el código de respuesta sea distinto de HTTP 200.

Este tipo de pruebas fueron aplicadas a las clases y métodos fundamentales implementados durante el desarrollo de la solución propuesta, lo cual permitió verificar el funcionamiento de las mismas y corregir los errores encontrados durante la etapa de implementación.

3.7 Pruebas de aceptación

Las pruebas de aceptación (PA) son una parte integral del desarrollo incremental definido por XP. Todas las historias de usuario son apoyadas por pruebas de aceptación, que son definidas por el cliente. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución (32).

La siguiente tabla muestra la especificación de una de las pruebas de aceptación diseñadas durante el desarrollo de la solución propuesta. Las pruebas restantes se encuentran en el **Anexo IV Pruebas de aceptación**.

Tabla 8 PA Consultar servicios del API REST

Prueba de aceptación		
Código: H.U1_P.A1	H.U: Implementar un API REST en el servidor de GRHS	
Nombre: Consultar servicios de API REST		
Descripción: Se realiza una solicitud HTTP a alguno de los servicios disponibles en el API-REST, para los datos del sistema que estos proveen.		
Condiciones de Ejecución: El servidor debe estar ejecutándose y el usuario debe poseer un usuario y una contraseña validos en alguno de los dominios del sistema.		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none">• Se construye la solicitud HTTP al servidor, definiendo los parámetros usuario y token de seguridad.• Se realiza la solicitud al servidor.		
Resultados esperados:	Escenario positivo	Escenario negativo
	Se muestran los datos solicitados al servidor.	Se muestra una página o un código de error.
Evaluación: Satisfactoria		

Dado que una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación y que la responsabilidad recae en todos los miembros del equipo de desarrollo, es recomendable publicar los resultados de las mismas, de manera que todo el equipo esté al tanto de esta información (32). Con este objetivo se resume, en el siguiente gráfico, los resultados de las pruebas de aceptación durante cada iteración del proceso de desarrollo de la solución propuesta.

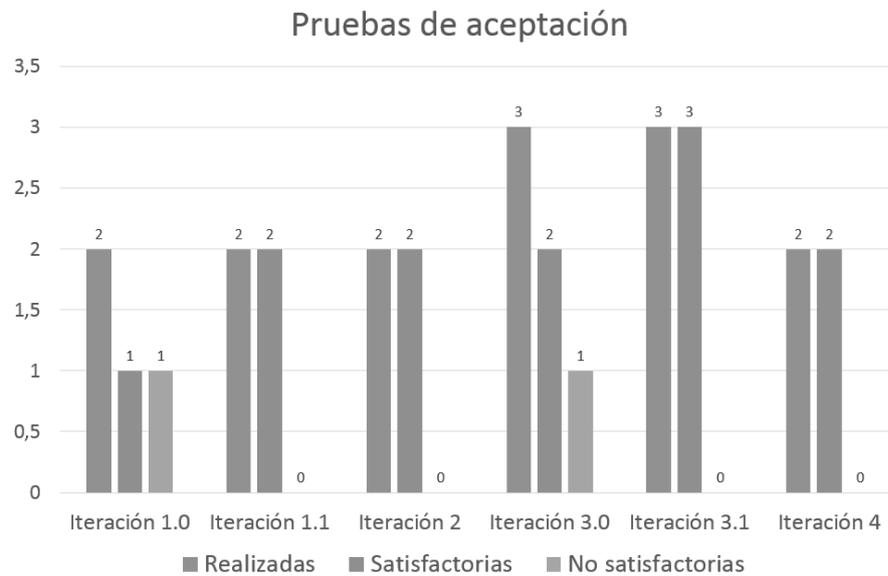


Figura 19 Resultados de las pruebas de aceptación.

Al final de una iteración de desarrollo, una HU es aceptada por el cliente si la prueba de aceptación correspondiente a la misma resulta satisfactoria. Si una prueba de aceptación falla, se realiza otra iteración de pruebas donde, tanto la HU como la prueba de aceptación correspondiente pueden sufrir cambios (33).

Al realizar las pruebas de aceptación, se detectaron no conformidades en dos de las iteraciones de desarrollo, por lo que se debió realizar dos iteraciones de pruebas extra. Esto implicó el rediseño de las pruebas de aceptación correspondientes y cambios en el código de la aplicación. Para verificar que dichos cambios no hubiesen afectado las funcionalidades que ya habían sido validadas, las iteraciones de prueba adicionales incluyeron, además de las pruebas de aceptación de las HU que habían fallado, la repetición de las pruebas de aceptación que se habían realizado en la iteración anterior y habían sido exitosas. Al finalizar la cuarta iteración no se detectaron no conformidades, con lo que se dio fin al proceso de pruebas.

Conclusiones del capítulo

En el presente capítulo se expuso el proceso de implementación de la solución propuesta, el cual dio lugar a un producto que cumple con las pautas definidas en el modelo descrito en el Capítulo 2. Se describieron las pruebas diseñadas para validar la implementación, tanto del API REST como de Gadmin Android, que sirvieron como elemento clave para validar que la solución cumple con las expectativas del cliente.

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 herramientas de gestión de recursos de hardware y software existentes y los enfoques de desarrollo de aplicaciones móviles, 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 desde dispositivos móviles. Esto permitió al equipo de desarrollo validar la premisa inicial de la presente investigación.
- Las etapas de planificación, diseño, implementación y prueba, definidas por la metodología de desarrollo de software seleccionada, facilitaron el cumplimiento del 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 de una aplicación Android para la administración del sistema GRHS permitió que dicho sistema contara con una interfaz visual para dispositivos móviles, lo cual facilita su uso y aumenta la competitividad del mismo como un potencial producto a ser adquirido por otras instituciones.
- Esta nueva aplicación abre las puertas a un gran número de posibilidades de expansión del alcance de GRHS como solución de software, al enlazar la gestión de recursos de hardware y software con las plataformas móviles.

Referencias bibliográficas

1. **Robinson, Kevin.** Wi-Fi and the Rise of Smart Cities. *Wi-Fi Alliance*. [En línea] 30 de Octubre de 2015. [Citado el: 13 de Enero de 2016.] <http://www.wi-fi.org/beacon/kevin-robinson/wi-fi-and-the-rise-of-smart-cities>.
2. **Rodríguez, Adrian E. Mena, y otros.** *Gestor de Recursos de Hardware y Software*. Departamento de Aplicaciones, Centro de Telemática de la Universidad de las Ciencias Informáticas. 2015. Presentado en el Encuentro Nacional de Seguridad en las Tecnologías de la Información 2015.
3. **Softinventive Lab Inc.** *Softinventive | Lab [documentation center]*. [En línea] 2016. http://docs.softinventive.com/En/Total_Network_Inventory_3/White_paper.
4. **NetSupport Limited.** *NetSupport DNA*. [En línea] 2016. <http://www.netsupportdna.com/ES/corporate/features.asp#Mobile>.
5. **OCS Inventory NG.** OCS Inventory NG | Demo. *OCS Inventory NG*. [En línea] 2016. [Citado el: 20 de Febrero de 2016.] <http://demo.ocsinventory-ng.org/>.
6. **GLPI.** Demo. *GLPI Free it and asset management*. [En línea] 2016. [Citado el: 30 de Enero de 2016.] <http://demo.glpi-project.org/front/central.php#>.
7. —. *GLPI Free it and asset management*. [En línea] 2016. [Citado el: 30 de Enero de 2016.] http://www.glpi-project.org/spip.php?page=annonce&id_breve=353&lang=en.
8. **Bateman, Sawyer.** Native or Responsive? : A look at the business case for each approach to mobile development. *UX Magazine*. [En línea] 2015. [Citado el: 30 de 11 de 2015.] <https://uxmag.com/articles/native-or-responsive>. ISSN: 2168 5681.
9. **Jobe, William.** *Native Apps vs. Mobile Web Apps*. Stockholm : Stockholm University, 2013.
10. **Letelier Ph.D. , Patricio y Penadés Ma., Carmen.** *Métodologías Ágiles para el Desarrollo de Software: eXtreme Programming (XP)*. Buenos Aires : Técnica Administrativa, Buenos Aires, 2006. Vol. 05. ISSN: 1666 1680.
11. **Gartner Inc.** *Gartner*. [En línea] 2016. [Citado el: 16 de Febrero de 2016.] <http://www.gartner.com/newsroom/id/2823619>.
12. **Universidad de las Ciencias Informáticas.** *Metodología de Desarrollo para la Actividad Productiva de la UCI*. La Habana : s.n.
13. **Oracle Corporation.** OpenJDK - JDK 7. *OpenJDK*. [En línea] 2015. [Citado el: 1 de Diciembre de 2015.] Copyright © 1995-2016, Oracle and/or its affiliates. All rights reserved.. <http://openjdk.java.net/projects/jdk7/>.

14. **Android Open Source Project.** Android Developers - API Levels. [En línea] 2015. [Citado el: 8 de 12 de 2015.] <https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels>.
15. **Django Software Foundation.** Django. [En línea] 2012. [Citado el: 8 de 12 de 2015.] <https://djangoproject.com/>.
16. **Android Open Source Project.** Android Developers - Connecting to the Network. *Android Developers*. [En línea] 2016. [Citado el: 3 de Febrero de 2016.] <https://developer.android.com/training/basics/network-ops/connecting.html>.
17. —. Android Developers - Creating a Sync Adapter. *Android Developers*. [En línea] 2016. [Citado el: 12 de Abril de 2016.] <https://developer.android.com/training/sync-adapters/creating-sync-adapter.html>.
18. —. Android Developers - Creating a Stub Authenticator. *Android Developers*. [En línea] 2016. [Citado el: 12 de Febrero de 2016.] <https://developer.android.com/training/sync-adapters/creating-authenticator.html>.
19. —. Android Developers - Content Provider Basics. *Android Developers*. [En línea] 2016. [Citado el: 12 de Febrero de 2016.] <https://developer.android.com/guide/topics/providers/content-provider-basics.html>.
20. **Beck, Kent.** *Extreme Programming Explained: Embrace Change*. New York : Addison-Wesley, 2000. Longman Inc..
21. **Cohn, Thomas y Paul, Ravi.** A Comparison of Requirements Engineering in Extreme Programming (XP) and Conventional Software Development Methodologies. s.l. : AMCIS 2001 Proceedings, 2001. Paper 256.
22. **Eberlein, Armin y Leite, J.C.S.P.** Agile Requirements Definition: A View from Requirements Engineering. *Proceedings of the International Workshop on Time-Constrained Requirements Engineering*. 2002. págs. 4-8.
23. **Len Bass, Paul Clements, Rick Kazman.** *Software Architecture in Practice*. New York : Addison-Wesley, 2003. ISBN: 0-321-15495-9 .
24. **Microsoft.** *Microsoft Application Architecture Guide*. Segunda. s.l. : Microsoft Press , 2009. © 2016 Microsoft. ISBN: 9780735627109.
25. **Larman, Craig.** *UML y Patrones, Una Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado*. Segunda. s.l. : Prentice Hall. págs. 163-217.
26. **National Instruments Corporation.** National Instruments: Patrones de diseño de aplicaciones: Maestro/Esclavo. *National Instruments*. [En línea] 7 de 10 de 2015. [Citado el: 15 de 4 de 2016.] <http://www.ni.com/white-paper/3022/es/>.

27. **Ambler, Scott.** *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process.* [ed.] Theresa Hudson . New York : Wiley Computer Publishing, 2002. Copyright © 2002 by Scott Ambler. All rights reserved. . ISBN: 0-471-20282-7.
28. **Beck, Kent, y otros.** *Manifiesto for Agile Software Development.* 2001.
29. **Oracle Corporation.** *Oracle Technology Network.* [En línea] Abril de 1999. [Citado el: 14 de Marzo de 2016.] Copyright © 1995-2016, Oracle and/or its affiliates. All rights reserved. . <http://www.oracle.com/technetwork/java/codeconventions-135099.html>.
30. **Python Software Foundation.** *Python.* [En línea] <https://www.python.org/dev/peps/pep-0008/>.
31. **Alchin, Marty.** *Pro Python.* New York : Apress, 2010. págs. 283-297. ISBN-13: 978-1-4302-2757-1.
32. **Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming.* s.l. : Universidad de Vigo, 2008. pág. 22.
33. **Finsterwalder, Malte.** Automating Acceptance Tests for GUI Applications in an Extreme Programming Environment. *Proceedings of the 2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering.* 2001. págs. 114-117.

Bibliografía

Alchin, Marty. *Pro Python*. New York : Apress, 2010. págs. 283-297. ISBN-13: 978-1-4302-2757-1.

Ambler, Scott. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. [ed.] Theresa Hudson . New York : Wiley Computer Publishing, 2002. Copyright © 2002 by Scott Ambler. All rights reserved. . ISBN: 0-471-20282-7.

Android Open Source Project. *Android Developers* . [En línea] 2015. <http://developer.android.com>.

Bateman, Sawyer. Native or Responsive? : A look at the business case for each approach to mobile development. *UX Magazine*. [En línea] 2015. [Citado el: 30 de 11 de 2015.] <https://uxmag.com/articles/native-or-responsive>. ISSN: 2168 5681.

Beck, Kent. *Extreme Programming Explained: Embrace Change*. New York : Addison-Wesley, 2000. Longman Inc..

Beck, Kent, y otros. *Manifiesto for Agile Software Development*. 2001.

Blé Jurado, Carlos, y otros. *Diseño Ágil con Test Driven Development*. Primera. 2010. Este libro se ha publicado bajo la Licencia Creative Commons. ISBN: 978-1-4452-6471-4.

Charland, Andre y Leroux, Brian. *Mobile Application Development: Web vs. Native*. 5, 2011, ACM Queue, Vol. 54.

Christie, Tom. Django REST framework 2. [En línea] Copyright (c) 2011-2014, Tom Christie, 2014. [Citado el: 2 de 12 de 2015.] <http://tomchristie.github.io/rest-framework-2-docs/>.

Cohn, Thomas y Paul, Ravi. A Comparison of Requirements Engineering in Extreme Programming (XP) and Conventional Software Development Methodologies. s.l. : AMCIS 2001 Proceedings, 2001. Paper 256.

Django Software Foundation. Django. [En línea] 2012. [Citado el: 8 de 12 de 2015.] <https://djangoproject.com/>.

Duque, Raúl González. *Python para todos*. 2011. págs. 6-7. Este libro se distribuye bajo una licencia Creative Commons Reconocimiento 2.5 España.

Eberlein, Armin y Leite, J.C.S.P. Agile Requirements Definition: A View from Requirements Engineering. *Proceedings of the International Workshop on Time-Constrained Requirements Engineering*. 2002. págs. 4-8.

Finsterwalder, Malte. Automating Acceptance Tests for GUI Applications in an Extreme Programming Environment. *Proceedings of the 2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering*. 2001. págs. 114-117.

- Gardner, Brett S.** *Responsive Web Design: Enriching the User Experience*. 1, s.l. : Noblis, 2011, Sigma Digital Ecosystem, Vol. 11. Copyright © 2011 Noblis, Inc., All Rights Reserved.
- GLPI.** Demo. *GLPI Free it and asset management*. [En línea] 2016. <http://demo.glpi-project.org/>.
- Gartner Inc.** *Gartner*. [En línea] 2016. [Citado el: 16 de Febrero de 2016.] <http://www.gartner.com/newsroom/id/2823619>.
- Hernández León, Rolando Alfredo y Coello González, Sayda.** *Metodología de la Investigación Científica*. La Habana : Editorial Universitaria, 2011. ISBN 978-959-16-1307-3.
- Jobe, William.** *Native Apps vs. Mobile Web Apps*. Stockholm : Stockholm University, 2013.
- Joskowicz, José.** *Reglas y Prácticas en eXtreme Programming*. s.l. : Universidad de Vigo, 2008. pág. 22.
- LACCEI.** Two-Column Template for LACCEI Conference Proceedings Based on IEEE Format. *14th LACCEI International Multi-Conference for Engineering, Education, and Technology: Engineering Innovations for Global Sustainability*. San José, Costa Rica : s.n., 20-22 de Julio de 2016.
- Larman, Craig.** *UML y Patrones, Una Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado*. Segunda. s.l. : Prentice Hall. págs. 163-217.
- Len Bass, Paul Clements, Rick Kazman.** *Software Architecture in Practice*. New York : Addison-Wesley, 2003. ISBN: 0-321-15495-9 .
- Letelier Ph.D. , Patricio y Penadés Ma., Carmen.** *Métodologías Ágiles para el Desarrollo de Software: eXtreme Programming (XP)*. Buenos Aires : Técnica Administrativa, Buenos Aires, 2006. Vol. 05. ISSN: 1666 1680.
- Microsoft.** *Microsoft Application Architecture Guide*. Segunda. s.l. : Microsoft Press , 2009. © 2016 Microsoft. ISBN: 9780735627109.
- National Instruments Corporation.** National Instruments: Patrones de diseño de aplicaciones: Maestro/Esclavo. *National Instruments*. [En línea] 7 de 10 de 2015. [Citado el: 15 de 4 de 2016.] <http://www.ni.com/white-paper/3022/es/>.
- Neil, Theresa.** *Mobile Design Pattern Gallery: UI Patterns for Smartphone Apps*. [ed.] Mary Treseler. Segunda. s.l. : O'Reilly Media, Inc., 2014. Copyright © 2016 Safari Books Online. ISBN 0636-9200-29311.
- NetSupport Ltd.** NetSupport DNA Getting Started Guide. 2012. Copyright© 2015 NetSupport Ltd All rights reserved .
- NetSupport Ltd.** NetSupport DNA Mobile Console Getting Started Guide . 2012. Copyright© 2012 NetSupport Ltd All rights reserved .

NetSupport Limited. *NetSupport DNA.* [En línea] 2016. <http://www.netsupportdna.com/ES/corporate/features.asp#Mobile>.

OCS Inventory NG. OCS Inventory NG | Demo. *OCS Inventory NG.* [En línea] 2016. [Citado el: 20 de Febrero de 2016.] <http://demo.ocsinventory-ng.org/>.

OCS Inventory NG. *OCS Inventory NG.* [En línea] 2016. <http://www.ocsinventory-ng.org/en/#functionalities-en>.

Oracle Corporation. OpenJDK - JDK 7. *OpenJDK.* [En línea] 2015. [Citado el: 1 de Diciembre de 2015.] Copyright © 1995-2016, Oracle and/or its affiliates. All rights reserved.. <http://openjdk.java.net/projects/jdk7/>.

Oracle Corporation. *Oracle Technology Newtwork.* [En línea] Abril de 1999. [Citado el: 14 de Marzo de 2016.] Copyright © 1995-2016, Oracle and/or its affiliates. All rights reserved. . <http://www.oracle.com/technetwork/java/codeconventions-135099.html>.

Python Software Foundation. *Python.* [En línea] <https://www.python.org/dev/peps/pep-0008/>.

R. M., Ken Auer. *Extreme Programming Applied.* New York : Addison-Wesley Professional, 2001.

Robinson, Kevin. Wi-Fi and the Rise of Smart Cities. *Wi-Fi Alliance.* [En línea] 30 de Octubre de 2015. [Citado el: 13 de Enero de 2016.] <http://www.wi-fi.org/beacon/kevin-robinson/wi-fi-and-the-rise-of-smart-cities>.

Rodríguez, Adrian E. Mena, y otros. *Gestor de Recursos de Hardware y Software.* Departamento de Aplicaciones, Centro de Telemática de la Universidad de las Ciencias Informáticas. 2015. Presentado en el Encuentro Nacional de Seguridad en las Tecnologías de la Información 2015.

Softinventive Lab Inc. *Softinventive | Lab [documentation center].* [En línea] 2016. http://docs.softinventive.com/En/Total_Network_Inventory_3/White_paper.

Softinventive Lab Inc. *Total Network Inventory.* [En línea] 2016. <http://www.softinventive.es/total-network-inventory/>.

Universidad de las Ciencias Informáticas. *Metodología de Desarrollo para la Actividad Productiva de la UCI.* La Habana : s.n.

Anexos

Anexo I Historias de Usuario

Tabla 9 HU-2 Implementar un mecanismo de manejo de permisos para el acceso al API REST

Historia de usuario	
Número: HU-2	Nombre de la historia de usuario: Implementar un mecanismo de manejo de permisos para el acceso al API REST
Modificación de la historia de usuario: 3	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración Asignada: 1
Prioridad en el negocio: Alta	Puntos Estimados: 1
Descripción: Cuando un usuario de GRHS realiza una solicitud a alguno de los servicios del API, se debe verificar cuáles son los privilegios del mismo en el sistema antes de devolver los resultados. Si no pertenece a la categoría Súper-Usuario, se deben filtrar los resultados que se le muestran de acuerdo a las ubicaciones que está autorizado a visualizar.	
Observaciones: Este filtro no aplica al servicio que muestre las alarmas del sistema, ya que las mismas no están relacionadas lógicamente con una ubicación en la implementación actual de gserver .	
Prototipo de interfaz: No procede	

Tabla 10 HU-3 Implementar un mecanismo de gestión de cuenta de usuario

Historia de usuario	
Número: HU-3	Nombre de la historia de usuario: Implementar un mecanismo de gestión de cuenta de usuario
Modificación de la historia de usuario: 2	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración Asignada: 1
Prioridad en el negocio: Alta	Puntos Estimados: 1
Descripción: Cuando un usuario va a crear una cuenta nueva en Gadmin Android se le muestra una pantalla con un formulario de autenticación. Los campos del formulario deben incluir entradas para el nombre de usuario, la contraseña, el dominio de autenticación y la dirección del servidor. Esta pantalla es la primera que se muestra la primera vez que se ejecuta la aplicación.	
Observaciones: Si se está modificando una cuenta existente, el campo nombre de usuario debe aparecer estático (no modificable) y el resto de los campos, excepto el de contraseña, completados con los datos de la cuenta existente.	
Prototipo de interfaz:	

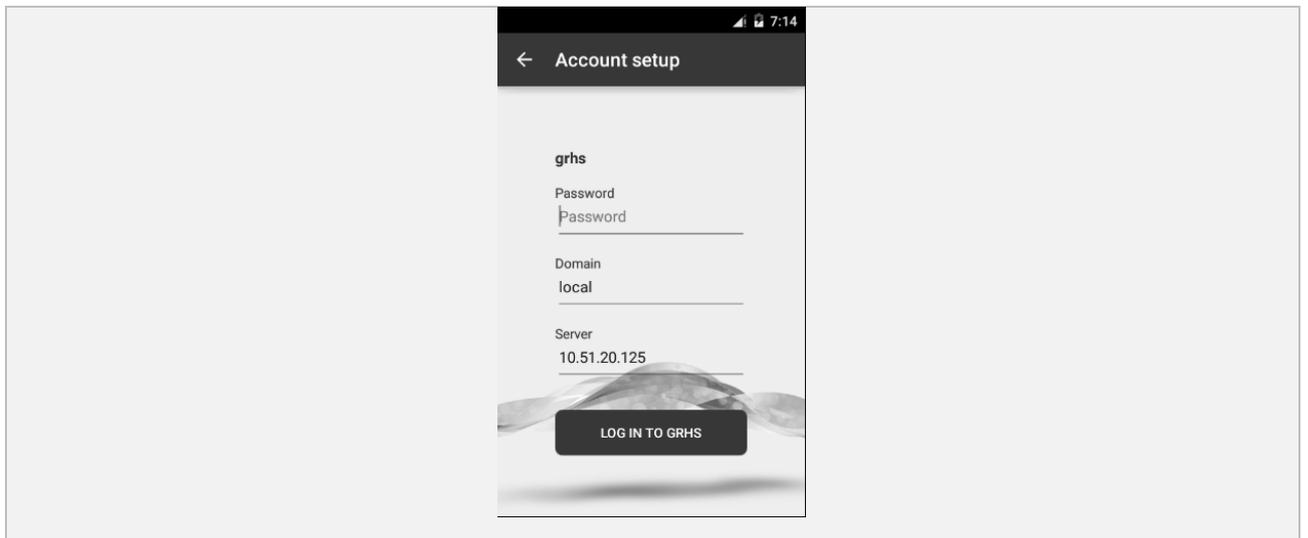


Tabla 11 HU-4 Implementar un mecanismo de descarga y sincronización de datos a demanda

Historia de usuario	
Número: HU-4	Nombre de la historia de usuario: Implementar un mecanismo de descarga y sincronización de datos a demanda
Modificación de la historia de usuario: Ninguna	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración Asignada: 2
Prioridad en el negocio: Alta	Puntos Estimados: 1
<p>Descripción: Debe permitir descargar los datos solicitados por el usuario. Este mecanismo comprueba si los datos ya existen en la base de datos local del dispositivo, en caso de que existan los actualiza, en caso contrario, los inserta. Antes de realizar una sincronización, la aplicación debe verificar si el porcentaje de espacio de almacenamiento libre del dispositivo es mayor que el parámetro especificado en las preferencias de la aplicación, si dicha condición se cumple, se procede con la sincronización, en caso contrario, se lanza una advertencia al usuario para que libere espacio en el dispositivo o cambie sus preferencias.</p>	
Observaciones: Ninguna	
Prototipo de interfaz:	

Tabla 12 HU-5 Mostrar estadísticas del sistema

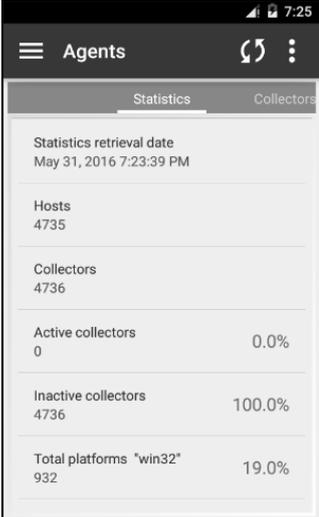
Historia de usuario	
Número: HU-5	Nombre de la historia de usuario: Mostrar estadísticas del sistema
Modificación de la historia de usuario: 2	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración Asignada: 2
Prioridad en el negocio: Alta	Puntos Estimados: 1
Descripción: Cuando un usuario navega a alguna de las categorías principales de navegación, o al inventario de algún tipo de activo específico, la primera pantalla que se muestra corresponde a las estadísticas correspondientes a esa categoría específica. Encabezando cada pantalla de estadísticas se encuentra la fecha en la que las mismas fueron obtenidas del servidor.	
Observaciones: Ninguna	
Prototipo de interfaz:	
	

Tabla 13 HU-6 Mostrar información de agentes, inventario, acciones, incidencias

Historia de usuario	
Número: HU-6	Nombre de la historia de usuario: Mostrar información de agentes, inventario, acciones, incidencias
Modificación de la historia de usuario: 3	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración asignada: 3
Prioridad en el negocio: Alta	Puntos estimados: 1

Descripción: La aplicación cuenta con vistas que contienen los listados de agentes, recursos de hardware y software, acciones e incidencias registradas en el servidor de GRHS. También ofrece vistas detalladas de cada uno de los objetos de dichas listas. Al entrar en la vista detallada de algún activo, si se toca el identificador del medio básico al que pertenece (Id del Colector), se muestra la pantalla con el inventario detallado del mismo.

Observaciones: Ninguna

Prototipo de interfaz:

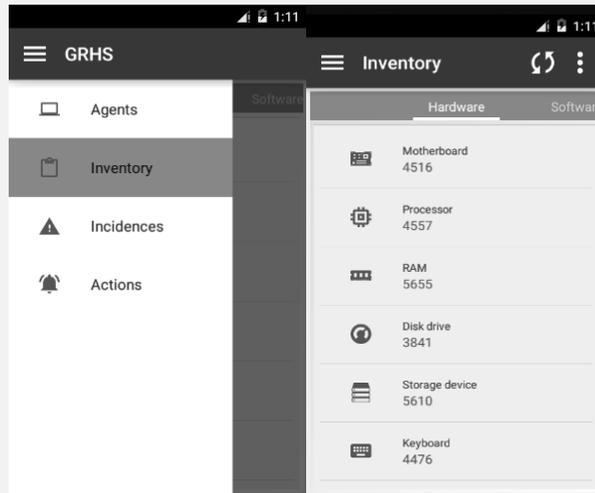


Tabla 14 HU-7 Implementar una interfaz para el manejo de preferencias de la aplicación

Historia de usuario	
Número: HU-7	Nombre de la historia de usuario: Implementar una interfaz para el manejo de preferencias de la aplicación
Modificación de la historia de usuario: Ninguna	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración Asignada: 3
Prioridad en el negocio: Alta	Puntos Estimados: 1
Descripción: Es una pantalla que permite la configuración de las preferencias de la aplicación. La misma debe ser accesible en todo momento, desde cualquier pantalla de la aplicación.	
Observaciones: Ninguna	
Prototipo de interfaz:	

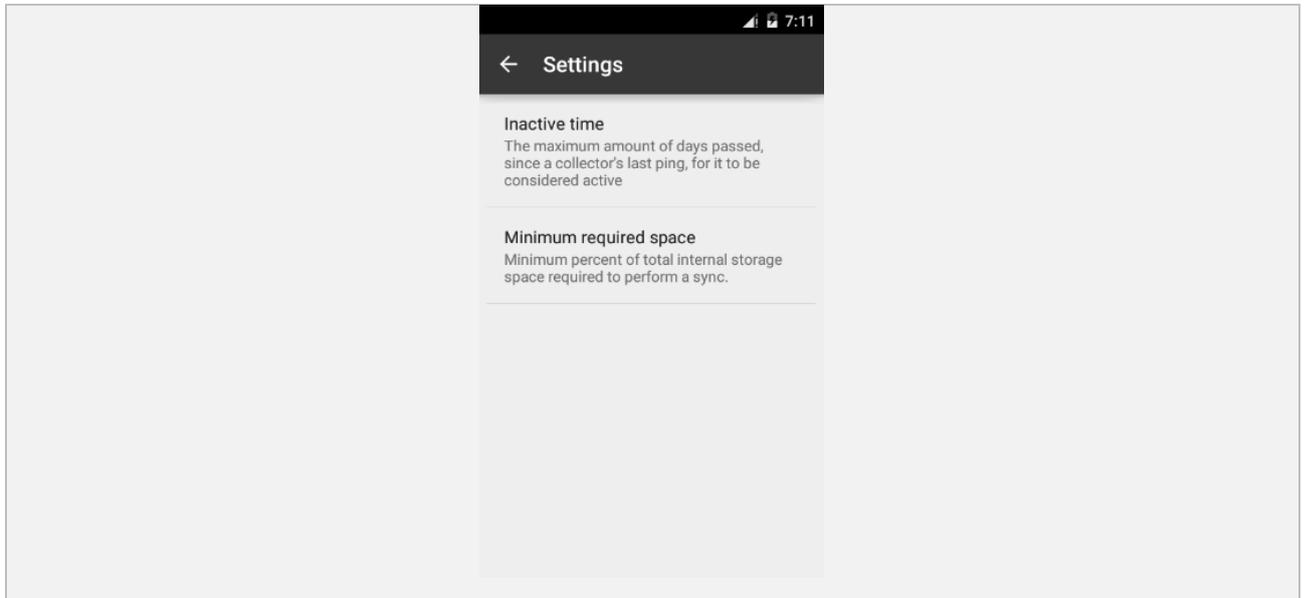


Tabla 15 HU-8 Gestionar la conexión de la aplicación mediante el protocolo HTTPS

Historia de usuario	
Número: HU-8	Nombre de la historia de usuario: Gestionar la conexión de la aplicación mediante el protocolo HTTPS
Modificación de la historia de usuario: Ninguna	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración Asignada: 3
Prioridad en el negocio: Alta	Puntos Estimados: 1
Descripción: Todas las solicitudes al servidor deben realizarse mediante el protocolo HTTPS, en pos de garantizar la seguridad de las credenciales que se incluyen en las mismas. Para ello deben implementarse los mecanismos correspondientes en cada clase encargada de la consulta y descarga de datos del servidor desde Gadmin Android.	
Observaciones: Ninguna	
Prototipo de interfaz: No procede	

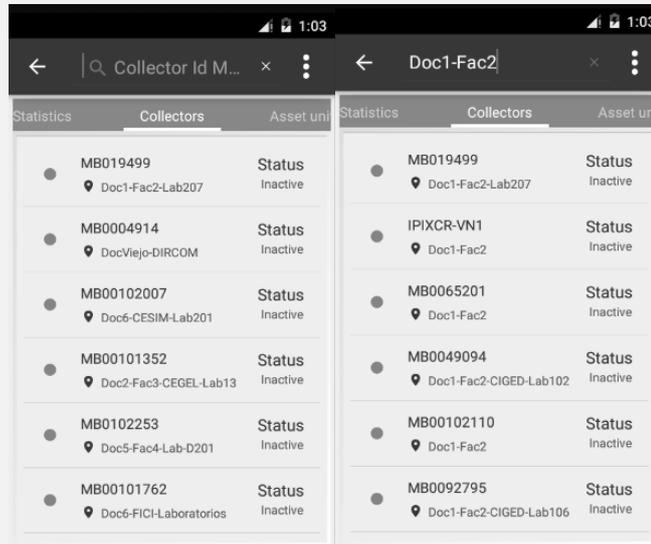
Tabla 16 HU-9 Implementar un mecanismo de búsqueda

Historia de usuario	
Número: HU-9	Nombre de la historia de usuario: Implementar un mecanismo de búsqueda
Modificación de la historia de usuario: Ninguna	
Usuario: Julia Roxana García Andalia, Nestor Antonio Castro Castillo	Iteración Asignada: 3
Prioridad en el negocio: Alta	Puntos Estimados: 1

Descripción: Crear un componente visual que permita la introducción de parámetros de búsqueda para filtrar los datos que se muestran en la aplicación, así como los datos que se descargan en las sincronizaciones manuales.

Observaciones: Ninguna

Prototipo de interfaz:



Anexo II Tarjetas CRC

Tabla 17 Tarjeta CRC AuthenticationActivity

Nombre de la clase: AuthenticationActivity

Responsabilidades de la clase: Clase llamada por AccountAuthenticator que genera una interfaz visual para que el usuario ingrese a su cuenta o se registre. Esta debe interactuar con el servidor para obtener el token y retornarlo al AccountAuthenticator.

Colaboradores de la clase:

- BaseAuthenticationActivity
- MainActivity
- R
- CustomSSLSocketFactory
- CustomX509TrustManager
- Constants
- NetUtils

Tabla 18 Tarjeta CRC AuthenticationService

Nombre de la clase: AuthenticationService

<p>Responsabilidades de la clase: Define un servicio enlazado con AccountAuthenticator que permite que el autenticador esté disponible en todo momento, tanto para la aplicación en cuestión, como para aplicaciones externas que deseen consultarlo.</p>
<p>Colaboradores de la clase:</p> <ul style="list-style-type: none"> • AccountAuthenticator

Tabla 19 Tarjeta CRC DataBaseContentProvider

<p>Nombre de la clase: DataBaseContentProvider</p>
<p>Responsabilidades de la clase: Es el componente que actúa como envoltura de seguridad para los datos de la aplicación, encapsulando la implementación SQLite de la base de datos local a través de un conjunto de URIs.</p>
<p>Colaboradores de la clase:</p> <ul style="list-style-type: none"> • DataBaseHelper • DataBaseContract

Tabla 20 Tarjeta CRC DataBaseHelper

<p>Nombre de la clase: DataBaseHelper</p>
<p>Responsabilidades de la clase: Provee los mecanismos básicos para la relación entre la aplicación Android y la base de datos local. Maneja la creación y actualización del esquema de la base de datos.</p>
<p>Colaboradores de la clase:</p> <ul style="list-style-type: none"> • DataBaseContract

Tabla 21 Tarjeta CRC DataBaseContract

<p>Nombre de la clase: DataBaseContract</p>
<p>Responsabilidades de la clase: Define el esquema de creación de la base de datos local mediante clases estáticas y constantes. Contiene las definiciones de URI de contenido para el manejo de los datos mediante un ContentProvider.</p>
<p>Colaboradores de la clase:</p> <ul style="list-style-type: none"> • UriMatcher • Uri • BaseColumns

Tabla 22 Tarjeta CRC SyncAdapter

Nombre de la clase: SyncAdapter
Responsabilidades de la clase: Maneja la sincronización de los datos obtenidos del servidor en un hilo de ejecución separado de la interfaz de usuario, y realiza las operaciones sobre la base de datos local.
Colaboradores de la clase: <ul style="list-style-type: none"> • DataBaseContract • Constants • DateTimeUtil • MathUtils • ListAPITask • ListByldAPITask • RetrieveAPITask

Tabla 23 Tarjeta CRC SyncService

Nombre de la clase: SyncService
Responsabilidades de la clase: Define un servicio enlazado a SyncAdapter, que permite que las funcionalidades de sincronización estén accesibles en todo momento y se ejecuten en un hilo aparte del de la interfaz de usuario.
Colaboradores de la clase: <ul style="list-style-type: none"> • SyncAdapter

Anexo III Tareas de Ingeniería

Tabla 24 TI Implementar el servicio GetHostByld

Tareas de Ingeniería		
Número: 2	H.U: Implementar un API-REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetHostByld		
Tipo: Desarrollo	Puntos estimados: 1/5	
Fecha inicio: 23 de marzo, 2016	Fecha fin: 23 de marzo, 2016	
Responsable: Nestor Antonio Castro Castillo		
Descripción: Se implementará un servicio que devuelva los datos de un colector, dado su identificador, así como los datos del inventario activo del mismo (información de todos los componentes de hardware y software relacionados con el mismo).		

Tabla 25 TI Implementar el servicio GetUnifiedMB

Tareas de Ingeniería		
Número: 3	H.U: Implementar un API REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetUnifiedMB		
Tipo: Desarrollo	Puntos estimados: 1/5	
Fecha inicio: 24 de marzo, 2016	Fecha fin: 24 de marzo, 2016	
Responsable: Nestor Antonio Castro Castillo		
Descripción: Se implementará un servicio que devuelva los datos de unificación de medios básicos registrados en el servidor.		

Tabla 26 TI Implementar el servicio GetLocations

Tareas de Ingeniería		
Número: 4	H.U: Implementar un API REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetLocations		
Tipo: Desarrollo	Puntos estimados: 1/5	
Fecha inicio: 25 de marzo, 2016	Fecha fin: 25 de marzo, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementará un servicio que devuelva las ubicaciones registradas en el servidor y las subredes asociadas a cada ubicación.		

Tabla 27 TI Implementar el servicio GetIncidentes

Tareas de Ingeniería		
Número: 5	H.U: Implementar un API REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetIncidentes		
Tipo: Desarrollo	Puntos estimados: 1/5	
Fecha inicio: 28 de marzo, 2016	Fecha fin: 28 de marzo, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementará un servicio que devuelva los datos de las incidencias registradas en el servidor.		

Tabla 28 TI Implementar el servicio GetStats

Tareas de Ingeniería

Número: 6	H.U: Implementar un API REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetStats		
Tipo: Desarrollo	Puntos estimados: 2/5	
Fecha inicio: 29 de marzo, 2016	Fecha fin: 30 de marzo, 2016	
Responsable: Nestor Antonio Castro Castillo		
Descripción: Se implementará un servicio que devuelva las estadísticas del sistema, separadas por categorías (agentes, hardware, software, incidencias, acciones).		

Tabla 29 TI Implementar el servicio GetInactiveTime

Tareas de Ingeniería		
Número: 7	H.U: Implementar un API REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetInactiveTime		
Tipo: Desarrollo	Puntos estimados: 1/5	
Fecha inicio: 31 de marzo, 2016	Fecha fin: 31 de marzo, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementará un servicio que devuelva el parámetro de configuración de tiempo de inactividad vigente en el servidor.		

Tabla 30 TI Implementar el servicio GetAlarms

Tareas de Ingeniería		
Número: 8	H.U: Implementar un API REST en el servidor de GRHS	Iteración: 1
Nombre: Implementar el servicio GetAlarms		
Tipo: Desarrollo	Puntos estimados: 1/5	
Fecha inicio: 1 de abril, 2016	Fecha fin: 1 de abril, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementará un servicio que devuelva los datos de las alarmas registradas en el servidor.		

Tabla 31 TI Implementar filtros de ubicaciones

Tareas de Ingeniería		
Número: 9	H.U: Implementar un mecanismo de manejo de permisos para el acceso al API REST	Iteración: 1
Nombre: Implementar filtros de ubicaciones		

Tipo: Desarrollo	Puntos estimados: 3/5
Fecha inicio: 4 de abril, 2016	Fecha fin: 6 de abril, 2016
Responsable: Nestor Antonio Castro Castillo	
Descripción: Implementar un filtro para las solicitudes en cada servicio del API REST que, dado el usuario que realiza la solicitud, devuelva solo los datos asociados a las ubicaciones que el mismo esté autorizado a visualizar.	

Tabla 32 TI Integrar el sistema de autenticación del API REST con los servidores LDAP

Tareas de Ingeniería		
Número: 10	H.U: Implementar un mecanismo de manejo de permisos para el acceso al API REST	Iteración: 1
Nombre: Integrar el sistema de autenticación del API REST con los servidores LDAP		
Tipo: Desarrollo	Puntos estimados: 2/5	
Fecha inicio: 7 de abril, 2016	Fecha fin: 8 de abril, 2016	
Responsable: Nestor Antonio Castro Castillo		
Descripción: Implementar un mecanismo que permita la autenticación en el API mediante los servidores LDAP configurados en el sistema GRHS. Para ello se obtendrán los datos del usuario que realiza la solicitud y se autorizará el acceso si pertenece al dominio especificado en la solicitud y la contraseña proporcionada es válida.		

Tabla 33 TI Implementar un Account Authenticator

Tareas de Ingeniería		
Número: 11	H.U: Implementar un mecanismo de gestión de cuenta de usuario	Iteración: 2
Nombre: Implementar un Account Authenticator		
Tipo: Desarrollo	Puntos estimados: 1	
Fecha inicio: 11 de abril, 2016	Fecha fin: 15 de abril, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementará un Account Authenticator, el cuál es básicamente un componente que provee Android para manejar las cuentas de usuario de la aplicación de manera profesional y segura. Su implementación es necesaria para poder utilizar posteriormente un componente Sync Adapter para gestionar la sincronización de los datos. Otros beneficios que aporta este componente son: una forma estándar de autenticar a los usuarios, simplifica autenticación para el desarrollador, manejo de casos de acceso denegado, compartir cuentas entre aplicaciones, entre otros. Implica la creación de un servicio de autenticación y de la interfaz visual de configuración de la cuenta.		

Tabla 34 TI Implementar un Content Provider

Tareas de Ingeniería		
Número: 12	H.U: Implementar un mecanismo de descarga y sincronización de datos a demanda	Iteración: 2
Nombre: Implementar un Content Provider		
Tipo: Desarrollo	Puntos estimados: 1	
Fecha inicio: 18 de abril, 2016	Fecha fin: 22 de abril, 2016	
Responsable: Nestor Antonio Castro Castillo		
<p>Descripción: Se implementará un Content Provider, componente de una aplicación Android que actúa como una envoltura de seguridad para los datos de la misma, son la interfaz estándar que conecta datos en un proceso con el código en ejecución en otro proceso. Esto implica la creación de una clase Contract para definir el esquema de creación de la base de datos local y una clase Helper para manejar la creación u actualización de la misma. La clase Content Provider contendrá los métodos necesarios para realizar operaciones sobre la base de datos.</p>		

Tabla 35 TI Implementar un Sync Adapter

Tareas de Ingeniería		
Número: 13	H.U: Implementar un mecanismo de descarga y sincronización de datos a demanda	Iteración: 2
Nombre: Implementar un Sync Adapter		
Tipo: Desarrollo	Puntos estimados: 1	
Fecha inicio: 25 de abril, 2016	Fecha fin: 29 de abril, 2016	
Responsable: Nestor Antonio Castro Castillo		
<p>Descripción: Se implementará un Sync Adapter, componente de la aplicación que encapsulará el código para transferir datos entre el dispositivo y la aplicación. Dicho componente se ejecutará en un hilo separado del hilo de ejecución de la interfaz de usuario, atado a un servicio que permitirá la disponibilidad del mismo en todo momento.</p>		

Tabla 36 TI Implementar vistas de estadísticas del sistema

Tareas de Ingeniería		
Número: 14	H.U: Mostrar estadísticas del sistema	Iteración: 3
Nombre: Implementar vistas de estadísticas del sistema		
Tipo: Desarrollo	Puntos estimados: 1	
Fecha inicio: 2 de mayo, 2016	Fecha fin: 6 de mayo, 2016	

Responsable: Julia Roxana García Andalia
Descripción: Se implementarán interfaces visuales que muestren las estadísticas del sistema según el contexto de la aplicación donde se encuentre el usuario (agentes, inventario, incidencias, acciones).

Tabla 37 TI Implementar vistas de listados de objetos por cada categoría

Tareas de Ingeniería		
Número: 15	H.U: Mostrar información de agentes, inventario, acciones, incidencias	Iteración: 3
Nombre: Implementar vistas de listados de objetos por cada categoría		
Tipo: Desarrollo	Puntos estimados: 2/5	
Fecha inicio: 9 de mayo, 2016	Fecha fin: 10 de mayo, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementarán interfaces visuales que muestren el listado de los elementos descargados del servidor (colectores, ubicaciones, alarmas, entre otros). Dichas listas se refrescarán por medio de componentes Cursor Loader, que escuchan las notificaciones del servicio de sincronización y actualizan las listas cuando el contenido de la base de datos cambia.		

Tabla 38 TI Implementar vistas de detalles de cada objeto

Tareas de Ingeniería		
Número: 16	H.U: Mostrar información de agentes, inventario, acciones, incidencias	Iteración: 3
Nombre: Implementar vistas de detalles de cada objeto		
Tipo: Desarrollo	Puntos estimados: 3/5	
Fecha inicio: 11 de mayo, 2016	Fecha fin: 13 de mayo, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementarán interfaces visuales que muestren los detalles de los objetos mostrados en una lista al tocarlos. La vista de detalles se comportará como un sistema de paginado, de manera que si se desean ver los detalles del objeto siguiente o el anterior, simplemente se realiza un gesto de barrido en la pantalla en la dirección de la lista en que se quiera desplazar el usuario. Esto evita tener que navegar de vuelta a la lista de objetos.		

Tabla 39 TI Implementar la actividad de preferencias

Tareas de Ingeniería

Número: 17	H.U: Implementar una interfaz para el manejo de preferencias de la aplicación	Iteración: 3
Nombre: Implementar la actividad de preferencias		
Tipo: Desarrollo	Puntos estimados: 1	
Fecha inicio: 16 de mayo, 2016	Fecha fin: 20 de mayo, 2016	
Responsable: Julia Roxana García Andalia		
Descripción: Se implementará una actividad de preferencias que permita la gestión de las configuraciones de la aplicación.		

Tabla 40 TI Modificar las clases que manejan la conexión al servidor para que empleen el protocolo de transferencia HTTPS

Tareas de Ingeniería		
Número: 18	H.U: Gestionar la conexión de la aplicación mediante el protocolo HTTPS	Iteración: 4
Nombre: Modificar las clases que manejan la conexión al servidor para que empleen el protocolo de transferencia HTTPS		
Tipo: Desarrollo	Puntos estimados: 1	
Fecha inicio: 23 de mayo, 2016	Fecha fin: 27 de mayo, 2016	
Responsable: Nestor Antonio Castro Castillo		
Descripción: Se implementarán un conjunto de clases para manejar parámetros de la conexión segura tales como los certificados de confianza, las autoridades certificadoras y los verificadores de "hostname".		

Tabla 41 TI Implementar un mecanismo de búsqueda

Tareas de Ingeniería		
Número: 19	H.U: Implementar un mecanismo de búsqueda	Iteración: 4
Nombre: Implementar un mecanismo de búsqueda		
Tipo: Desarrollo	Puntos estimados: 1	
Fecha inicio: 30 de mayo, 2016	Fecha fin: 3 de junio, 2016	
Responsable: Nestor Antonio Castro Castillo		
Descripción: Se implementará un mecanismo que permita realizar búsquedas sobre las listas de datos que se muestran en la aplicación, mediante la comparación de cadenas de texto y el uso de filtros de búsqueda. Dicho mecanismo permitirá realizar búsquedas filtradas sobre los objetos de la base de datos local, así como especificar parámetros de búsqueda específicos para las solicitudes de sincronización.		

Esto permitirá que los resultados que se descarguen del servidor, durante una sincronización hecha mientras hay filtros activos, sean aquellos que coincidan con los filtros especificados.

Anexo IV Pruebas de aceptación

Tabla 42 PA Verificar el funcionamiento del mecanismo de manejo de permisos para el acceso al API REST

Prueba de aceptación		
Código: H.U2_P.A2	H.U: Implementar un mecanismo de manejo de permisos para el acceso al API REST	
Nombre: Verificar el funcionamiento del mecanismo de manejo de permisos para el acceso al API REST		
Descripción: Se realiza una solicitud a alguno de los servicios del API REST, para que liste activos ubicados en una localización determinada, de la cual se conoce previamente que el usuario no tiene permiso para verla.		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> El servidor está ejecutándose. 		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none"> Se construye la solicitud HTTP al servidor. Se ejecuta la solicitud. 		
Resultados esperados:	Escenario positivo	Escenario negativo
	El servidor devuelve una página con 0 elementos.	El servidor devuelve una página con 1 o más elementos.
Evaluación: Satisfactoria		

Tabla 43 PA Crear una cuenta de usuario nueva

Prueba de aceptación		
Código: H.U3_P.A3	H.U: Implementar un mecanismo de gestión de cuenta de usuario	
Nombre: Crear una cuenta de usuario nueva		
Descripción: Mostrar el formulario de autenticación al usuario. Al llenar los campos y validar los datos de entrada, se solicita el token de seguridad al servidor y se crea una nueva cuenta.		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> La aplicación se encuentra en ejecución. 		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none"> Se muestra un formulario de autenticación con los campos a llenar. Se introducen los datos requeridos en el formulario sin dejar campos vacíos. Se acciona el botón de autenticación. Se solicita el token de seguridad al servidor. Se crea la nueva cuenta. Se inicia la actividad principal. 		

Resultados esperados:	Escenario positivo	Escenario negativo
	Se crea una nueva cuenta, iniciándose la actividad principal y la primera sincronización.	Muestra un mensaje de error.
Evaluación: Satisfactoria		

Tabla 44 PA Sincronización manual

Prueba de aceptación		
Código: H.U4_P.A4	H.U: Implementar un mecanismo de descarga y sincronización de datos a demanda	
Nombre: Sincronización manual		
Descripción: Se solicita una sincronización de los datos a la clase SyncAdapter mediante el servicio de sincronización, verificando previamente que la red está activa y que hay espacio libre en el dispositivo.		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> • Existe una conexión activa a la red Wi-Fi. • Existe el mínimo espacio libre requerido para realizar una sincronización. • Las credenciales de autenticación son válidas. 		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none"> • Realizar la solicitud de los datos al servicio correspondiente. • Serializar los datos utilizando los modelos definidos para ello en la aplicación. • Verificar si los datos locales necesitan ser actualizados. • Insertar los datos nuevos. 		
Resultados esperados:	Escenario positivo	Escenario negativo
	Se inicia el proceso de sincronización y se actualiza la base de datos y las vistas asociadas.	Muestra un mensaje de error.
Evaluación: Satisfactoria		

Tabla 45 PA Mostrar estadísticas del sistema

Prueba de aceptación		
Código: H.U5_P.A5	H.U: Mostrar estadísticas del sistema	
Nombre: Mostrar estadísticas del sistema		
Descripción: Ejecuta alguna de las interfaces de estadísticas de la aplicación.		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> • Existe una cuenta de usuario creada en el sistema. • Se ha realizado al menos una sincronización del servicio de estadísticas de manera satisfactoria. 		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none"> • Ejecutar la aplicación. 		

<ul style="list-style-type: none"> Navegar hacia la primera vista de cualquiera de las 4 categorías principales (Agentes, Inventario, Incidencias, Estadísticas). 		
Resultados esperados:	Escenario positivo	Escenario negativo
	Muestra las estadísticas del sistema correspondientes al contexto de la aplicación en que se encuentra el usuario.	Muestra el texto “No hay resultados que mostrar”.
Evaluación: Satisfactoria		

Tabla 46 PA Mostrar información de agentes, inventario, acciones, incidencias

Prueba de aceptación		
Código: H.U6_P.A6	H.U: Mostrar información de agentes, inventario, acciones, incidencias	
Nombre: Mostrar información de agentes, inventario, acciones, incidencias		
Descripción: Mostrar la lista y los detalles de los diferentes tipos de elementos descargados del servidor (colectores, recursos de hardware y software, incidencias y alarmas).		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> Existe una cuenta de usuario creada en el sistema. Se ha realizado al menos una sincronización exitosa del servicio que devuelve la información que se desea mostrar. 		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none"> Ejecutar la aplicación. Navegar hacia alguna de las vistas que muestran listas de objetos descargados. 		
Resultados esperados:	Escenario positivo	Escenario negativo
	Se muestra la lista de objetos descargados del servidor, así como la información detallada de los mismos al tocarlos.	Muestra el texto “No hay resultados que mostrar”.
Evaluación: Satisfactoria		

Tabla 47 PA Configurar las preferencias de la aplicación

Prueba de aceptación		
Código: H.U7_P.A7	H.U: Implementar una interfaz para el manejo de preferencias de la aplicación	
Nombre: Configurar las preferencias de la aplicación		
Descripción:		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> Existe una cuenta de usuario creada en el sistema. 		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none"> Ejecutar la aplicación. Abre el panel desplegable de la barra de acción. Seleccionar la opción “Preferencias”. 		

<ul style="list-style-type: none"> • Cambiar el valor por defecto de la preferencia, por ejemplo: cambiar la preferencia “Tamaño de página”, de 10 a 25. 		
Resultados esperados:	Escenario positivo	Escenario negativo
	El cambio de configuración es aplicado correctamente: se descargan 25 objetos en la siguiente sincronización, en lugar de 10.	El cambio de configuración no se aplica: se descargan solo 10 objetos en la siguiente sincronización.
Evaluación: Satisfactoria		

Tabla 48 PA Realizar una solicitud de autenticación al servidor mediante una conexión HTTPS

Prueba de aceptación		
Código: H.U8_P.A8	H.U: Gestionar la conexión de la aplicación mediante el Protocolo HTTPS	
Nombre: Realizar una solicitud de autenticación al servidor mediante una conexión HTTPS		
Descripción: Se intenta realizar la creación de una cuenta nueva en la aplicación, que implica la solicitud del token de seguridad al servidor, mediante una conexión HTTPS.		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> • El servidor se está ejecutando y aceptando conexiones mediante el protocolo HTTPS por el puerto 443. • Existe una conexión activa a la Wi-Fi en el dispositivo. 		
Entrada / Pasos de ejecución:		
<ul style="list-style-type: none"> • Iniciar la actividad de autenticación de la aplicación. • Llenar el formulario de autenticación • Ejecutar la solicitud. 		
Resultados esperados:	Escenario positivo	Escenario negativo
	Se realiza una conexión exitosa al servidor y se crea la nueva cuenta (asumiendo que las credenciales sean las correctas).	Muestra un mensaje de error.
Evaluación: Satisfactoria		

Tabla 49 PA Ejecutar una búsqueda filtrada

Prueba de aceptación		
Código: H.U9_P.A9	H.U: Implementar un mecanismo de búsqueda	
Nombre: Ejecutar una búsqueda filtrada		
Descripción: Se introducen palabras clave en el campo de búsqueda de la barra de acción de la aplicación, la lista entonces modifica su contenido de acuerdo a dichos parámetros. Además, las sincronizaciones realizadas mientras la búsqueda esté activa adecuarán los parámetros para solicitar al servidor sólo los objetos que cumplan con los mismos.		
Condiciones de Ejecución:		
<ul style="list-style-type: none"> • La vista activa actualmente en la aplicación es un listado de objetos. 		

- La barra de búsqueda está desplegada.

Entrada / Pasos de ejecución:

- Introducir la cadena de texto o los filtros de deseados
- Presionar el botón de ejecución de la búsqueda

Resultados esperados:	Escenario positivo	Escenario negativo
	Si existen datos que coincidan con los parámetros de búsqueda, la lista se actualiza para mostrar solo esos datos.	La lista no se modifica.

Evaluación: Satisfactoria