Universidad de las Ciencias Informáticas

Facultad 1

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

Título:

Soporte de comunicación *proxy* para sistemas Android

Autores:

Laura Pérez Vera

Jorge Alberto Díaz Orozco

Tutores:

MsC. Allan Pierra Fuentes

Ing. Dairelys García Rivas

Consultante:

Ing.Yaiselis Ramírez Mastrapa

La Habana, Cuba

Junio, 2015

"[...] ¿ Para qué, sino para poner paz entre los hombres han de ser los adelantos de la ciencia?"

José Martí

Declaración de autoría

Declaramos ser los únicos autores de este trabajo y se autoriza a la Universidad de las Ciencias Informáticas a que haga el uso que estime pertinente con el mismo. Para que así conste firmamos la presente a los ___ días del mes de _____ del año 2015.

_____ _____

Jorge Alberto Díaz Orozco Laura Pérez Vera

_____ _____

Ing. Dairelys García Rivas MsC. Allan Pierra Fuentes

Dedicatoria

De Laura:

A mis abuelos que no están hoy aquí conmigo, por el amor y la confianza que siempre tuvieron en mí. A mis padres por ser mis guías incansables y mi fuente de energía. A mi hermana por ser mi mejor amiga y por dibujarme una sonrisa en todo momento.

De Jorge:

A mis padres y hermana que dan sentido a la palabra familia…
A mi novia que es la dirección a donde apunta mi rosa de los vientos…
A los Elefantes Azules...

Agradecimientos

De Laura:

Para mis abuelos es mi primer agradecimiento y mi primera deuda aunque no se encuentren físicamente hoy conmigo, pero sé que me deben estar mirando donde quiera que estén. A mi mamá por ser la mujer más completa que conozco y por convertirme en la persona que soy, a mi papi por ser mi superhéroe, mi amor eterno y mi novio incondicional. A mi hermana, por ser mi mejor amiga y mi cómplice. A cada persona en mi familia, por el amor y el apoyo. A Orozco por ser mi jefe sin plantilla, por ser mi amigo y compartir estos momentos. A Dayrelis, Allan y Yaiselis por la ayuda y la paciencia. A las amistades hechas durante este tiempo, en especial a Rogelio y Lisandra por la ayuda y consejos para seguir, a Betty por compartir momentos difíciles e inolvidables. A Neyvis por brindarme más que una amistad por mucho tiempo, por ser de mi familia. A Jorge Reyes por estar cuando lo necesitaba. A todas las personas que ayudaron a que este trabajo sea lo que hoy es.

De Jorge:

A mis padres, porque sin ellos nada hubiera sido posible. A mi hermanita, mi flacucha, que siempre ha estado junto a mí. A toda mi familia que me ha demostrado que es en la unión y el amor donde se puede encontrar la felicidad. A mi novia por amarme sin límites y no darse por vencida. A mis suegros que me acogieron como a un hijo. A Laura, mi jefa incansable y amiga jovial, que ha sido ejemplo de paciencia y esmero durante tanto tiempo. A nuestros tutores y consultante, que nos guiaron en este camino limando las asperezas de nuestro trabajo. A todos los profesores y amigos que contribuyeron a nuestra formación y pusieron un pedazo de sí para ayudarnos. A mis amigos, seres especiales que siempre están ahí para mí sin importar las condiciones. A la comunidad de Software Libre y el colectivo de humanOS por todos estos años de trabajo y amistad. A Ale y toda la alquimia que da forma a los Elefantes Azules…

Resumen

El desarrollo que ha tenido la telefonía celular en los últimos años trajo consigo la aparición de los teléfonos inteligentes. Estos teléfonos se encuentran construidos sobre plataformas informáticas, siendo Android una de las más utilizadas y su uso altamente demandado en Cuba. La conexión a Internet constituye una de las funcionalidades que más buscan los usuarios cuando utilizan un teléfono inteligente, pero en Cuba se realiza mediante *proxies*. Ellos ofrecen soporte de autenticación basados en diferentes protocolos, siendo NTLM uno de los más usados. Android no trae por defecto la autenticación de usuarios para la conexión a Internet mediante *proxies*, es por eso que requiere de aplicaciones adicionales que resuelvan esta situación. Luego de analizar los conceptos relacionados con la investigación, se realizó un estudio de las diferentes tecnologías que llevan a cabo la autenticación en la conexión a Internet mediante *proxies* en Android. Como resultado del análisis se obtuvo que es necesario la creación de una aplicación que sea capaz de satisfacer esta funcionalidad, con los protocolos de autenticación NTLM y NTLMv2 y que no necesite permisos de administración para su correcto funcionamiento. Finalmente se realizaron pruebas al sistema que permitieron validar que el producto final está listo para su uso.

Palabras claves: NTLM, NTLMv2, *proxy*.

Índice de contenido

Introducción....1

Capítulo 1. Fundamentación teórica....7

1.1.Conceptos asociados a la investigación....7
1.1.1.Teléfonos inteligentes....7
1.1.2.Servidor proxy....7
1.1.3.Protocolos de autenticación....8
1.1.3.1.Kerberos....8
1.1.3.2.NTLM y NTLMv2....10
1.2.Tecnologías utilizadas por Android para conectarse a los proxies....11
1.2.1.Comparación entre tecnologías utilizadas por Android para conectarse a los proxies....12
1.2.1.1.Autoproxy....12
1.2.1.2.ProxyDroid....12
1.2.1.3.SandroProxy....13
1.2.2.Resultado de la comparación entre las tecnologías utilizadas por Android para conectarse a un proxy....13
1.3.Tecnologías utilizadas....14
1.3.1.Metodología de desarrollo....14
1.3.2.Entorno de Desarrollo Integrado....15
1.3.3.Lenguaje Unificado de Modelado....16
1.3.4.Herramienta CASE....17
1.3.5.Lenguajes de Programación....17
1.3.6.Android SDK v18....19
1.3.7.Java-NTLM-Proxy v0.2....19
1.3.8.JCIFS v1.3.17....19
1.4.Conclusiones parciales....20

Capítulo 2. Análisis y diseño de la solución propuesta....21

2.1.Propuesta de solución....21
2.2.Desglose de responsabilidades....22
2.3.Listado de requisitos funcionales y no funcionales....24
2.4.Análisis del dominio....27
2.4.1.Modelo de dominio....27
2.4.2.Diagrama de clases (Ver Anexo 2)....28
2.5.Patrones utilizados en el desarrollo de la solución....28
2.5.1.Patrones GoF....29
2.5.2.Patrones Generales de Software para la Asignación de Responsabilidades....31

2.6.Descripción de los actores del sistema....32
2.7.Diagrama de casos de uso del sistema....33
2.7.1.Patrones de casos de uso utilizados....33
2.8.Descripción de los casos de uso del sistema....34
2.9.Conclusiones parciales....37

Capítulo 3. Implementación y realización de pruebas....38

3.1.Implementación....38
3.1.1.Estándar de codificación....39
3.1.2.Diagrama de componentes....39
3.1.3.Diagrama de despliegue....40
3.2.Pruebas de software....40
3.2.1.Pruebas de aceptación....40
3.2.1.1.Casos de pruebas de aceptación....41
3.2.1.2.Resultados de las pruebas de aceptación....42
3.3.Resultados del uso de UCIntlm....44
3.4.Conclusiones parciales....46

Conclusiones....47

Recomendaciones....48

Referencias Bibliográficas....49

Glosario de siglas y términos....51

Anexos....52

Anexo 1.Encuesta aplicada....52
Anexo 2. Diagrama de Clases....54

Índice de ilustraciones

Ilustración 1:Comparación entre plataformas móviles.....2
Ilustración 2: Proceso de autenticación NTLM.....11
Ilustración 3: Esquema de propuesta.....22
Ilustración 4: Modelo de dominio.....28
Ilustración 5: Patrón Abstract Factory.....30
Ilustración 6: Patrón Observer.....31
Ilustración 7: Diagrama de casos de uso del sistema.....33
Ilustración 8: Diagrama de Componentes.....39
Ilustración 9: Diagrama de Despliegue.....40
Ilustración 10: Resultado de las pruebas de Aceptación.....43
Ilustración 11: Estadística de descargas.....44
Ilustración 12: Frecuencia de uso.....45
Ilustración 13: Comparación en la encuesta.....45
Ilustración 14: Diagrama de Clases.....54

Índice de tablas

Tabla 1: Comparación entre tecnologías Autoproxy, ProxyDroid y SandroProxy.....14
Tabla 2: Tabla de Responsabilidades.....24
Tabla 3: Descripción de los actores del sistema.....33
Tabla 4: Modificaciones en fase de implementación.....38
Tabla 5: Escenarios de Prueba.....42

Introducción

En los últimos años la telefonía celular ha tenido un gran auge y desarrollo, actualmente es capaz de brindar otros servicios diferentes que para los que fue creada inicialmente, surgiendo así los teléfonos inteligentes. Estos se diferencian de los teléfonos convencionales en que buscan cada vez más igualarse en funcionalidades a una computadora, siendo la característica más importante que permiten la instalación de programas para incrementar el procesamiento de datos y la conectividad a Internet (Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, Deborah Estrin 2010).

Los teléfonos inteligentes están construidos sobre una plataforma informática que permite darle servicio a diferentes aplicaciones móviles, estas aplicaciones son diseñadas para ser ejecutadas en teléfonos inteligentes, tabletas y otros dispositivos móviles (Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, Deborah Estrin 2010).

La Ilustración 1 representa la tabla que contiene la cantidad de unidades vendidas (*Global Smartphone Operating System Shipments*) de los principales Sistemas Operativos (SO) de teléfonos inteligentes como son Android, *Apple iOS*, *Microsoft*, *Blackberry* y de otros, estableciendo una comparativa entre el segundo trimestre del año 2013 (Q2´13) y el segundo trimestre del año 2014 (Q2´14). La segunda parte muestra cómo se encuentran estos SO en el mercado (*Global Smartphone Operating System Marketshare*) para los mismos períodos de tiempo. En la parte final se muestra expresado en por ciento el crecimiento anual de los trimestres antes mencionados (*Total Growth Year over Year*) (Ken Hyers 2014).

Se evidencia del análisis de la tabla anterior que Android domina las ventas del mercado con un 80,2% en el segundo trimestre del 2013 y con un 84,6% en el segundo trimestre del 2014. Esto se debe a que se desarrolla de forma abierta, permitiendo acceder tanto al código fuente como a la lista de incidencias, donde se pueden ver problemas aún no resueltos y reportar problemas nuevos, además Android como *software* libre permite la distribución de las mejoras hechas en su código fuente de forma gratuita, logrando así que todos los usuarios que lo utilicen salgan beneficiados (Ken Hyers 2014).

Global Smartphone Operating System Shipments (Millions of Units)	Q2 '13	Q2 '14
Android	186.8	249.6
Apple iOS	31.2	35.2
Microsoft	8.9	8.0
BlackBerry	5.7	1.9
Others	0.5	0.5
Total	233.0	295.2

Global Smartphone Operating System Marketshare %	Q2 '13	Q2 '14
Android	80.2%	84.6%
Apple iOS	13.4%	11.9%
Microsoft	3.8%	2.7%
BlackBerry	2.4%	0.6%
Others	0.2%	0.2%
Total	100.0%	100.0%

Total Growth Year-over-Year %	48.9%	26.7%

Ilustración 1:Comparación entre plataformas móviles

En Cuba esto constituye una ventaja objetiva porque permite estudiar, modificar y mejorar su diseño mediante la disponibilidad de su código fuente, siendo esta la tecnología por la que apuesta el país en la nueva era de la informatización de la sociedad orientada a dispositivos inteligentes.

Una de las principales funcionalidades que utiliza un usuario en un dispositivo inteligente es la conexión a Internet, pero en Cuba esta conexión se encuentra principalmente a nivel institucional, donde se realiza mediante *proxies*.

“Un *proxy* es un intermediario que controla, administra y atiende las peticiones de conexión entre equipos de cómputo de una red local con la red de Internet. Es un dispositivo de *hardware* (equipo de cómputo, dispositivo electrónico) o de *software* (programa computacional) que realiza de manera controlada la comunicación entre los equipos que pertenecen a la red local en la que se encuentra dicho dispositivo y

un servidor web de cualquier parte del mundo con un dominio en Internet y que además sirve como control de seguridad en el acceso remoto a equipos de una red" (Carlos Aguilar Rodríguez 2010).

El trabajo con *proxies* se convierte en una herramienta fundamental para la administración de recursos informáticos en las empresas y centros educacionales. En el caso de Cuba es primordial porque permite el aseguramiento de la red local para la restricción del acceso a contenido inseguro, además de permitir establecer las diferentes políticas de acceso, brindar velocidad en las conexiones, así como el ahorro de recursos.

El *proxy* provee soporte para el acceso de los usuarios a la red, basado en diferentes protocolos de autenticación, entre los que se encuentran *Kerberos*, *Active Directory* y NTLM (*Network LAN Manager*), siendo NTLM la opción más común por los usuarios (Eric Glass 2006).

El SO Android no presenta soporte para la autenticación de usuarios en el acceso a Internet, requiere de aplicaciones adicionales que realicen esta funcionalidad. Actualmente, existen aplicaciones que tratan de resolver esta necesidad, tales como: ProxyDroid, Autoproxy y SandroProxy, pero ellas presentan deficiencias en su solución.

Dada la situación explicada anteriormente surge como **problema a resolver** ¿cómo garantizar el soporte para *proxy* con esquemas de autenticación NTLM y NTLMv2 de las aplicaciones del SO Android, para el acceso a los recursos de Internet?

El **objeto de estudio** en el cual se enmarca el problema planteado lo constituyen las bases tecnológicas[1] para el soporte de conexión a través de *proxies* con esquemas de autenticación, siendo su **campo de acción** las bases tecnológicas para el soporte de conexión a través de *proxies*, con esquemas de autenticación NTLM y NTLMv2, en los teléfonos inteligentes con el SO Android.

Para dar solución al problema planteado, se traza como **objetivo general** desarrollar la base tecnológica que garantice el soporte para *proxy* con esquemas de autenticación NTLM y NTLMv2 en el SO Android. Este se desglosa en los siguientes **objetivos específicos**:

1 Referido a los componentes de *software* que son necesarios generar cuando se realiza el proceso de conexión a Internet, dígase librerías, aplicaciones y configuraciones del sistema

- Elaborar el marco teórico conceptual relacionado con los aspectos teóricos que sustentan la investigación.
- Realizar el análisis de la solución planteada.
- Diseñar e implementar la solución al problema planteado.
- Validar la solución a partir de los diferentes escenarios.

Para dar cumplimiento al objetivo general, se definen las siguientes **tareas de investigación**:

- Elaboración de los fundamentos teóricos de la investigación.
- Revisión de la bibliografía asociada a los esquemas de autenticación NTLM y NTLMv2 así como soluciones desarrolladas anteriormente.
- Análisis y diseño de la solución.
- Definición de las herramientas a utilizar para la implementación de la solución.
- Diseño de los casos de prueba para la evaluación de la solución.
- Validación del resultado obtenido.
- Corrección de los posibles errores surgidos durante la validación.

Los autores del presente trabajo de diploma proponen como **idea a defender**, el desarrollo de la base tecnológica para el SO Android con soporte NTLM y NTLMv2, puede garantizar la conexión de las aplicaciones del SO hacia internet a través de servidores *proxies*.

Para dar cumplimiento a las tareas de investigación se emplearon los siguientes métodos.

Métodos Científicos:

- **Analítico-Sintético:** Utilizado en la fundamentación teórica, para la definición de cada una de las tareas propuestas que responden al estudio de los principales conceptos como son: protocolo de

autenticación NTLM y NTLMv2, *proxy* y su interrelación con los SO móviles, además fue utilizado en la subdivisión de las tareas relacionadas con la construcción de la solución propuesta.

- **Histórico-Lógico:** Utilizado en el estudio de las principales tecnologías ya existentes que intentan responder al problema planteado, pero que presentan deficiencias, mediante una investigación sobre su funcionamiento se pudo establecer sus ventajas y sus desventajas.
- **Análisis Documental:** Utilizado en el estudio de la documentación referente al problema planteado, para seleccionar aquellas ideas que fueran importantes señalar y analizar, permitiendo establecer un análisis objetivo de lo que se quiere estudiar.

Métodos empíricos:

- **Observación:** Utilizado en varios momentos de la investigación, inicialmente para definir la necesidad de una solución que funcione como *proxy* y resuelva la conectividad a Internet mediante los protocolos de autenticación NTLM y NTLMv2, además fue utilizado en las diferentes etapas para el desarrollo de la base tecnológica y en los métodos de prueba, permitiendo prevenir la repetición de errores y poder observar la respuesta de la comunidad a las diferentes versiones de la aplicación, hasta llegar al resultado final.

Métodos particulares:

- **Encuesta:** Utilizada en la fase final del desarrollo de la solución, para obtener las diferentes opiniones de los usuarios, así como la aceptación que tuvo en la comunidad. La encuesta estuvo formada por preguntas abiertas y cerradas en la obtención de información.

Como continuación de lo antes expuesto, se divide el estudio en tres capítulos:

El **primer capítulo** aborda el estudio de los principales conceptos asociados a la investigación, para lograr una mayor comprensión de la misma y de las tecnologías ya existentes que resuelven de forma parcial el problema planteado, como son Autoproxy, Proxydroid y Sandroproxy. Se realiza un estudio del arte de las

mismas, teniendo en cuenta sus ventajas y las razones por las cuales no son óptimas en su solución. Se establecen las tecnologías y herramientas que servirán para el posterior desarrollo de la solución.

El **segundo capítulo** aborda el diseño de la solución propuesta en la realización de los artefactos que define la metodología a utilizar, además se realiza el análisis y diseño de los diferentes casos de uso, basándose en el análisis previo de la problemática.

En el **tercer capítulo** se realiza la implementación de la solución planteada y el diseño de los casos de prueba que se van a aplicar a la misma, analizando los errores presentados y las opiniones de los diferentes usuarios.

Capítulo 1. Fundamentación teórica

En este capítulo se analizan los principales conceptos relacionados con el funcionamiento de los *proxies*, para la conectividad a Internet en el SO Android con los protocolos de autenticación NTLM y NTLMv2. Se realiza una comparación entre las principales aplicaciones que resuelven de forma parcial el problema de la investigación. Finalmente se establece una conclusión parcial acerca de los resultados de la investigación, sirviendo esta para el desarrollo de la solución dada.

1.1. Conceptos asociados a la investigación

1.1.1. Teléfonos inteligentes

Es un teléfono móvil construido sobre una plataforma informática, que permite darle soporte a las diferentes aplicaciones móviles. Estos dispositivos se diferencian de los teléfonos convencionales en que buscan cada vez más igualarse en funcionalidades a una computadora, siendo la característica más importante que permiten la instalación de programas para incrementar el procesamiento de datos y la conectividad (Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, Deborah Estrin 2010).

A nivel mundial se han desarrollado varias plataformas y entre las más usadas se encuentran Android (de *Google*), *iOS* (de *Apple*), *Symbian* (de *Nokia*), *BlackBerry OS* (de *BlackBerry*) y *Windows Phone* (de *Microsoft*) (Ken Hyers 2014).

1.1.2. Servidor *proxy*

Un servidor *proxy* es un equipo que actúa como intermediario entre los equipos de una red de área local e Internet mediante protocolos de autenticación, efectuando solicitudes a Internet en su lugar. De esta manera, cuando un usuario se conecta a Internet con una aplicación cliente configurada para utilizar un servidor *proxy*, la aplicación primero se conecta con el servidor *proxy* y le da la solicitud. El servidor *proxy* se vincula entonces al servidor al que la aplicación del cliente desea conectarse y le envía la solicitud,

después el servidor le envía la respuesta al *proxy*, el cual a su vez la envía a la aplicación del cliente (Greg Ferro 2014).

El uso de un servidor *proxy* presenta ventajas como son la disminución tanto en el uso de ancho de banda en Internet como en el tiempo de acceso a los documentos de los usuarios. Esto viene dado por la capacidad de los servidores *proxies* de guardar en memoria las páginas que los usuarios de la red de área local visitan comúnmente, para poder proporcionarlas lo más rápido posible. Por otra parte, al utilizar un servidor *proxy*, las conexiones pueden registrarse al crear registros de actividad (*logs*) para guardar sistemáticamente las peticiones de los usuarios cuando solicitan conexiones a Internet. Gracias a esto, las conexiones de Internet pueden filtrarse al analizar tanto las solicitudes del cliente como las respuestas del servidor (Greg Ferro 2014).

1.1.3. Protocolos de autenticación

Un protocolo de autenticación es un tipo de protocolo criptográfico que es utilizado con la finalidad de lograr la autenticación de entidades que desean comunicarse de forma segura. Existen diferentes protocolos de autenticación, entre los más usados se encuentran *Kerberos* y NTLM (Eric Glass 2006).

1.1.3.1. *Kerberos*

Protocolo de autenticación de redes creado por el Instituto Tecnológico de Massachusetts (MIT por las iniciales de su nombre en inglés, *Massachusetts Institute of Technology*) que permite a dos computadoras en una red insegura demostrar su identidad de manera segura. Sus diseñadores se concentraron primeramente en un modelo de cliente-servidor y brindar autenticación mutua: tanto cliente como servidor verifican la identidad uno del otro. Los mensajes de autenticación están protegidos para evitar *eavesdropping*[2] y ataques de *replay*[3] (Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schiller 2012).

2 Término en inglés que traducido al español significa escuchar secretamente. Se refiere a ataques de escuchas, tanto sobre medios con información cifrada, como no cifrada.

3 Forma de ataque de red en el cual una transmisión de datos válida es maliciosa o fraudulentamente repetida.

Un servidor *Kerberos* (KDC, por las iniciales de su nombre en inglés *Kerberos Distribution Center*) provee dos servicios fundamentales: el de autenticación (AS, por las iniciales de su nombre en inglés *Authentication Service*) y el de *tickets* (TGS, por las iniciales de su nombre en inglés *Ticket Granting Service*). El primero tiene como función autenticar inicialmente a los clientes y proporcionarles un *ticket* para comunicarse con el segundo, el servidor de *tickets*, que proporciona a los clientes las credenciales necesarias para comunicarse con un servidor final que es quien realmente ofrece un servicio. Además, el servidor posee una base de datos de sus clientes (usuarios o programas) con sus respectivas claves privadas, conocidas únicamente por dicho servidor y por el cliente al que pertenece (Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schiller 2012).

Kerberos mantiene una base de datos de claves secretas; cada entidad en la red —sea cliente o servidor — comparte una clave secreta conocida únicamente por él y *Kerberos*. El conocimiento de esta clave sirve para probar la identidad de la entidad. Para una comunicación entre dos entidades, *Kerberos* genera una clave de sesión, la cual pueden usar para asegurar sus problemas (Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schiller 2012).

Kerberos puede ser difícil de implementar por las siguientes razones: (Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schiller 2012)

- La migración de contraseñas de usuarios desde una base de datos de contraseñas estándar UNIX[4] a una base de datos de contraseñas *Kerberos*, puede ser complicada y no hay un mecanismo rápido para realizar esta tarea.
- *Kerberos* presupone que cada usuario es de confianza, pero que está utilizando una máquina no fiable en una red no fiable. Su principal objetivo es el de prevenir que las contraseñas no cifradas sean enviadas a través de la red, sin embargo, si cualquier otro usuario, aparte del usuario adecuado, tiene acceso a la KDC para la autenticación, *Kerberos* estaría en riesgo.
- Para que una aplicación use *Kerberos*, el código debe ser modificado para hacer las llamadas apropiadas a las librerías de *Kerberos*. Para algunas aplicaciones, esto puede suponer un esfuerzo excesivo de programación, debido al tamaño de la aplicación o su diseño. Para otras aplicaciones

4 Sistema Operativo portable, multitarea y multiusuario desarrollado en 1969.

incompatibles, los cambios se deben realizar en el modo en que el servidor de red y sus clientes se comunican; esto puede suponer bastante programación.

- Si se transmite cualquier contraseña a un servicio que no usa *Kerberos* para autenticar, existe el riesgo de que el paquete pueda ser interceptado.

1.1.3.2.NTLM y NTLMv2

NTLM (NT LAN *Manager*) o en su más reciente versión NTLMv2 es un protocolo de autenticación de la *suite Microsoft* que se basa en nombres de usuario y contraseñas cifradas antes de enviar esta información a través de la red. Las variables con las que trabaja NTLM se basan en los datos obtenidos durante el proceso de inicio de sesión interactivo y se componen de: un nombre de dominio, un nombre de usuario y una función resumen unidireccional de la contraseña del usuario. NTLM utiliza un protocolo de desafío/respuesta cifrado para autenticar a un usuario sin enviar la contraseña del usuario a través de la red (Andres Tarasco Acuña 2008).

Los pasos siguientes representan un esquema de autenticación NTLM:

1. El usuario accede a un equipo cliente y proporciona un nombre de dominio, nombre de usuario y contraseña. El equipo cliente calcula una función resumen criptográfica de la contraseña y descarta la contraseña real.

2. El cliente envía el nombre de usuario al servidor (en texto plano).

3. El servidor genera un número aleatorio de 16 bytes, llamado reto y lo envía al equipo cliente.

4. El cliente cifra este desafío con la función resumen de la contraseña del usuario y devuelve el resultado al servidor. Esto se conoce como la respuesta.

5. El servidor envía los siguientes tres elementos al controlador de dominio: nombre de usuario, desafío que envía al cliente y la respuesta recibida desde el cliente.

6. El controlador de dominio utiliza el nombre de usuario para recuperar el valor de la función resumen de la contraseña del usuario de la base de datos. Utiliza este valor de la contraseña para cifrar el desafío.

7. El controlador de dominio compara el desafío cifrado calculado (en el paso 6) con la respuesta calculada por el cliente (en el paso 4). Si son idénticas, la autenticación es exitosa.

NTLM es ampliamente utilizado en situaciones donde un controlador de dominio no está disponible o es inalcanzable. Por ejemplo, NTLM se utilizaría si un cliente no es capaz de trabajar con *Kerberos*, el servidor no está unido a un dominio o el usuario se autentica de forma remota a través de Internet (Andres Tarasco Acuña 2008).

Para mayor comprensión del funcionamiento de este protocolo se muestra la Ilustración 2.

Ilustración 2: Proceso de autenticación NTLM

1.2.Tecnologías utilizadas por Android para conectarse a los *proxies*

Las tecnologías en el SO Android que permiten conectarse a un *proxy* determinado que presente protocolo de autenticación NTLM o NTLMv2, para acceder a los recursos en Internet son varias, los autores del presente trabajo de diploma centraron su análisis en aquellas que intentaban responder en mayor medida al problema de la investigación, estas fueron: Autoproxy, ProxyDroid y SandroProxy.

1.2.1.Comparación entre tecnologías utilizadas por Android para conectarse a los *proxies*

A continuación se establece una comparación entre las principales tecnologías, teniendo en cuenta sus ventajas y sus desventajas, obteniendo así las características por las cuales son consideradas como candidatas para dar solución al problema.

1.2.1.1.Autoproxy

Esta tecnología funciona mediante la creación de un *proxy* que se ejecuta de manera transparente, que intercepta lo que se ejecuta en el dispositivo para redirigir el tráfico a su *proxy*. Todo el tráfico saliente se captura, se formatea y se transmite a través del *proxy* de su red (*Autoproxy* 2013).

Ventajas (*Autoproxy* 2013)

- Soporta los *proxies* HTTP, HTTPS, SOCKS4 y SOCKS5. (Squid, Forefront, Polipo, Tor, Apache, Privoxy, Tinyproxy, etc).
- Soporta autenticación básica, implícita: NTLM y NTLMv2.
- Posibilidad de múltiples configuraciones de *proxy*.
- Disponible en inglés, portugués, francés, italiano, español y alemán.

Desventajas (*Autoproxy* 2013)

- El dispositivo electrónico donde sea instalada debe trabajar con permiso de superusuario[5] y que tenga instalado previamente *iptables*[6] para garantizar que todo funcione de manera correcta.

1.2.1.2.ProxyDroid

Tecnología que permite la configuración del *proxy* en los dispositivos que utilicen Android (*ProxyDroid* 2014).

5 Nombre convencional de la cuenta de usuario que posee todos los derechos. Cuenta del administrador.

6 Aplicación que le permite a un administrador de sistema configurar las tablas, cadenas y reglas en el mismo.

Ventajas (*ProxyDroid* 2014)

- Soporte para los *proxies* HTTP / HTTPS / SOCKS4 / SOCKS5.
- Soporte básico para NTLM y NTLMv2.
- Funcionamiento de *proxy* individual para una o más aplicaciones.

Desventajas (*ProxyDroid* 2014)

- El dispositivo donde se dispone a ser instalada esta aplicación tiene que trabajar con permiso de superusuario.

1.2.1.3.SandroProxy

Tecnología para Android creada con el objetivo de mejorar la privacidad al navegar la web desde un dispositivo móvil (*SandroProxy* 2014).

Ventajas (*SandroProxy* 2014)

- Funciona con la versión de Android 2.3 o superior.

Desventajas (*SandroProxy* 2014)

- No presenta soporte para el protocolo de autenticación NTLMv2.

1.2.2.Resultado de la comparación entre las tecnologías utilizadas por Android para conectarse a un *proxy*

Como resultado del análisis anterior y a modo de resumen se muestra la Tabla 1 donde las anteriores tecnologías son comparadas en base a las características de soporte para el protocolo NTLM y NTLMv2, la necesidad de tener permisos de administrador en cada aplicación y la característica de código abierto que esclarece si la aplicación permite acceder a su código fuente.

Características/ Tecnologías	Soporte NTLM	Soporte NTLMv2	Necesidad de permisos de administrador	Código Abierto
Autoproxy	X	X	X	X
ProxyDroid	X	X	X	X
SandroProxy	X			X

Tabla 1: Comparación entre tecnologías Autoproxy, ProxyDroid y SandroProxy

Del análisis previo se concluye que actualmente no existe una solución que permita la conectividad a Internet mediante protocolos de autenticación NTLM o NTLMv2, para el SO Android de forma eficiente.

1.3.Tecnologías utilizadas

Como parte del desarrollo de *software* se definen herramientas y tecnologías que serán utilizadas para garantizar el óptimo resultado por parte del equipo de desarrollo. A continuación se definen el conjunto de tecnologías y herramientas necesarias.

1.3.1.Metodología de desarrollo

Uno de los principales aspectos de un proyecto de *software* es el proceso de desarrollo que se va seguir para la implementación de un producto. Dentro de este proceso es necesario especificar las actividades del ciclo de vida del *software*, esto viene dado por la metodología que sea usada en el mismo. Para el correcto desarrollo de la aplicación es necesario el uso de una metodología que sea capaz de conducir al equipo de trabajo, con el objetivo de asegurar la calidad y eficacia durante el proceso de concepción del producto (Sonia Pinzón, Juan Carlos Guevara Bolaños 2006).

OpenUP

El Departamento de Sistemas Operativos determinó el uso de la metodología OpenUp para el desarrollo de *software*, teniendo en cuenta que la misma está dirigida a la gestión y el desarrollo de proyectos de *software* basados en un desarrollo iterativo, ágil e incremental.

La metodología OpenUP define **cuatro fases** (*OpenUP* 2012):

1. Inicio: En esta fase, las necesidades de cada participante del proyecto son tomadas en cuenta y son plasmadas en los objetivos del proyecto. Se definen para el proyecto: el ámbito, los límites, el criterio de aceptación, los casos de uso críticos, una estimación inicial del coste y un boceto de la planeación.

2. Elaboración: En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Se debe elaborar un plan de proyecto, estableciendo unos requisitos y arquitectura estables. Al final de la fase se debe tener una definición clara y precisa de los casos de uso, actores, la arquitectura del sistema y un prototipo ejecutable.

3. Construcción: En esta fase todos los componentes y funcionalidades del sistema que falten por implementar son realizados, probados e integrados. Los resultados obtenidos en forma de incrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado.

4. Transición: Esta fase corresponde a la introducción del producto en la comunidad de usuarios. La fase de transición consta de las sub-fases de pruebas beta, pilotaje y capacitación de los usuarios finales y de los encargados del mantenimiento del sistema. En función a la respuesta obtenida por los usuarios, puede ser necesario realizar cambios en las entregas finales o implementar alguna funcionalidad más solicitada por la mayoría.

1.3.2.Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado, del inglés *Integrated Development Environment* (IDE), es un programa informático compuesto por un conjunto de herramientas de programación. Los IDE proveen un marco de trabajo para los lenguajes de programación (Daniel González, Miguel Hernández, José Garduño 2003).

Android *Studio* rc 1.0

Es un IDE para la plataforma Android. Se encuentra disponible para desarrolladores de forma gratuita. Es diseñado específicamente para desarrollar soluciones para Android y su uso es recomendado por Google,

razón por la cual es usada por los autores del presente trabajo de investigación (*Android Studio Overview | Android Developers* 2014).

Android *Studio* presenta las siguientes características (*Android Studio Overview | Android Developers* 2014):

- Renderización[7] en tiempo real.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción y estadísticas de uso.
- Refactorización[8] específica de Android y arreglos rápidos.
- Herramientas para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.

1.3.3.Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés *Unified Modeling Language*) es el lenguaje de modelado de sistemas de *software*, utilizado para visualizar, especificar, construir y documentar un sistema. Es aplicable en el proceso de construcción de un *software* para dar soporte a una metodología de desarrollo de *software*, ayudando a especificar, visualizar y documentar esquemas de sistemas de *software*, incluyendo su estructura y diseño, de una manera que cumpla con todos los requisitos del mismo. El uso de herramientas basadas en UML permite analizar los requisitos y diseñar una solución que sea satisfactoria (*Unified Modeling Language (UML)* 2014).

7 Término usado para referirse al proceso de generar una imagen o vídeo mediante el cálculo de iluminación partiendo de un modelo en 3D.

8 Técnica de la ingeniería de *software* para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

1.3.4.Herramienta CASE

Las herramientas CASE, acrónimo de *Computer Aided Software Engineering* (Ingeniería de *Software* Asistida por Computadoras) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software*, reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida del desarrollo del *software*, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras (Patricia López 2014).

***Visual Paradigm* v8.0**

Visual Paradigm es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de varios tipos de diagramas (Patricia López 2014).

1.3.5.Lenguajes de Programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por computadoras. Son utilizados para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático, se le llama programación (Robert W. Sebesta 2012).

En Android existen dos formas de manejar la implementación, ellas son mediante **código dalvik** o **código nativo**.

Código Dalvik

Dalvik es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. La Máquina Virtual Dalvik (DVM, por sus siglas en inglés Dalvik *Virtual Machine*) sacrifica la portabilidad que caracteriza a Java para poder crear aplicaciones con un mejor rendimiento y menor consumo de energía. Estas dos características son extremadamente importantes en dispositivos móviles, debido a que la capacidad de las baterías en estos dispositivos es limitada. Esta máquina virtual corre por encima de un *kernel*[9] Linux, el cual le permite, entre otras cosas, delegar las tareas relacionadas con la gestión de hilos y memoria a bajo nivel (*What is Android?: Android Runtime* 2006).

DVM permite ejecutar aplicaciones programadas en **Java**, siendo este un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo. Las aplicaciones de Java son generalmente compiladas a *bytecode* (clase Java) que puede ejecutarse en cualquier Máquina Virtual Java (JVM, por sus siglas en inglés Java *Virtual Machine*) sin importar la arquitectura de la computadora subyacente (Byous,Jon 2005).

Código Nativo

Es una forma de código de la programación de computadora que se configura para funcionar con el uso de un procesador especificado. La estructura exacta del código nativo se usa para responder a las instrucciones que son enviadas por el procesador (Sheng Liang 2000).

Siguiendo este concepto se encuentra **Java *Native Interface*** (**JNI**), siendo este un *framework* de programación que permite que un programa escrito en Java ejecutado en la JVM pueda interactuar con programas escritos en otros lenguajes como C, C++ y Ensamblador. El JNI se usa para escribir métodos nativos que permitan solventar situaciones en las que una aplicación no puede ser enteramente escrita en Java, como por ejemplo, en el caso de que la biblioteca estándar de clases no proporcione soporte para funcionalidades dependientes de la plataforma (Sheng Liang 2000).

9 *Software* que constituye una parte fundamental del sistema operativo. Es el principal responsable de facilitar a los distintos programas acceso seguro al *hardware* de la computadora o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

Teniendo en cuenta los conceptos anteriores, los autores del presente trabajo de investigación harán uso para el desarrollo de la solución del Código Dalvik mediante el trabajo con DVM, teniendo en cuenta que se encuentra optimizada para necesitar poca memoria, aprovechando así el mejor rendimiento de los móviles, además DVM se encuentra distribuida como *software* libre.

1.3.6. Android SDK v18

Android SDK siglas en inglés de *Software Development Kit* (*kit* de desarrollo de *software*) es un conjunto de herramientas de desarrollo de *software* que le permite al programador crear aplicaciones para un sistema concreto, en este caso para Android. Este paquete o *kit* de desarrollo incluye las API siglas en inglés de *Application Programing Interface* (Interfaz de Programación de Aplicaciones) y herramientas necesarias para desarrollar las aplicaciones, utilizando Java como lenguaje de programación (*Android Studio Overview* | *Android Developers* 2014).

1.3.7. Java-NTLM-Proxy v0.2

Para el desarrollo de la solución propuesta, se va a realizar un trabajo basado en la aplicación Java-NTLM-Proxy, tomando de la misma los elementos necesarios para el trabajo en la solución. Java-NTLM-Proxy es una aplicación que permite a clientes web que no tienen soporte para el esquema de autenticación de *proxy* NTLM de *Microsoft*, el acceso para conectarse a los servidores *proxies* de *Microsoft*. Esta aplicación permite editar las propiedades del *proxy*, estableciendo así el nombre de *host* y los detalles de la cuenta de dominio que va a utilizar para conectarse a Internet, además puede reenviar un puerto local a una máquina remota (*Java NTLM Proxy* 2007).

1.3.8. JCIFS v1.3.17

JCIFS del inglés *Java Common Internet File System* (Sistema de Archivos Común de Internet de Java) es una librería cliente de código abierto que implementa el SMB[10] del inglés *Server Message Block* (Servidor de Bloqueo de Mensajes). JCIFS es un protocolo de red CIFS (Sistema de Archivos de Internet Común)

10 Protocolo de red (que pertenece a la capa de aplicación en el modelo OSI) que permite compartir archivos e impresoras (entre otras cosas) entre nodos de una red. Es utilizado principalmente en ordenadores con *Microsoft Windows* y DOS.

desarrollado totalmente en Java. CIFS es el protocolo estándar de intercambio de archivos en la plataforma *Microsoft Windows*. Este cliente se utiliza ampliamente en la producción de grandes redes locales (*The Java CIFS Client Library* 2008).

Los autores del presente trabajo de investigación hicieron uso de esta tecnología, en la utilización del esquema de autenticación NTLMv2 y autenticación HTTP.

1.4.Conclusiones parciales

En este capítulo, con el objetivo de darle solución al problema planteado, se realizó un estudio de diferentes tecnologías que presenta Android para la conexión a Internet mediante *proxy*. El mismo permitió distinguir los siguientes aspectos, donde se obtuvo como resultado que no cumplen los requerimientos necesarios para darle solución a la problemática.

- Las tecnologías Autoproxy y ProxyDroid para su correcto funcionamiento el dispositivo tiene que trabajar con permiso de administrador.
- La tecnología SandroProxy no presenta soporte para el protocolo de autenticación NTLM y NTLMv2, y para su correcto funcionamiento el dispositivo tiene que trabajar con permiso de administrador.

Estas deficiencias encontradas permiten afirmar la necesidad de desarrollar una nueva base tecnológica que cuente con soporte para el protocolo de autenticación NTLM y NTLMv2 y que no requiera trabajar con permiso de administrador. Para lograr el desarrollo de esta base tecnológica se define el uso del lenguaje de programación Java, además de trabajar con el IDE Android *Studio* y con Android SDK.

Capítulo 2. Análisis y diseño de la solución propuesta.

En el presente capítulo se analiza y diseña la propuesta de solución del sistema a implementar, guiada por la metodología de desarrollo OpenUP. Además, se describen las características que debe cumplir el sistema a través del diseño de los diferentes casos de uso basados en requisitos funcionales y no funcionales.

2.1.Propuesta de solución

Se propone la creación de una base tecnológica que funcionará en dispositivos inteligentes que cuenten con el SO Android, permitiendo la conexión al *proxy* con autenticación NTLM y NTLMv2 para la navegación a Internet.

La solución propuesta funcionará como *proxy* de autenticación, sirviendo de intermediario entre el cliente que envía la solicitud y el *proxy*. Este intermediario se ocupará del mecanismo de autenticación en lugar del cliente. Los autores del presente trabajo de diploma utilizaron como base para el desarrollo de esta solución la aplicación *proxy* de escritorio Java-NTLM-Proxy, reutilizando parte de su código. Para mayor comprensión del funcionamiento de la solución propuesta se encuentra la Ilustración 3, que muestra inicialmente las interfaces y el *widget*[11] con las que el usuario interactúa, permitiendo que el mismo introduzca sus datos como nombre de usuario y contraseña, además de permitirle especificar opciones avanzadas de la configuración *proxy*, todo esto con la finalidad de iniciar el servicio. Una vez el usuario seleccione iniciar el servicio, Android inicia el servidor *proxy* y configura el sistema para su utilización, a su vez la solución utiliza el mecanismo de Java-NTLM-Proxy para el proceso de autenticación con NTLM, además de la librería JCIFS, estos mecanismos se conectan con el servidor *proxy* institucional, estableciendo así la conexión a Internet.

11 Pequeña aplicación o programa que tiene como objetivo dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

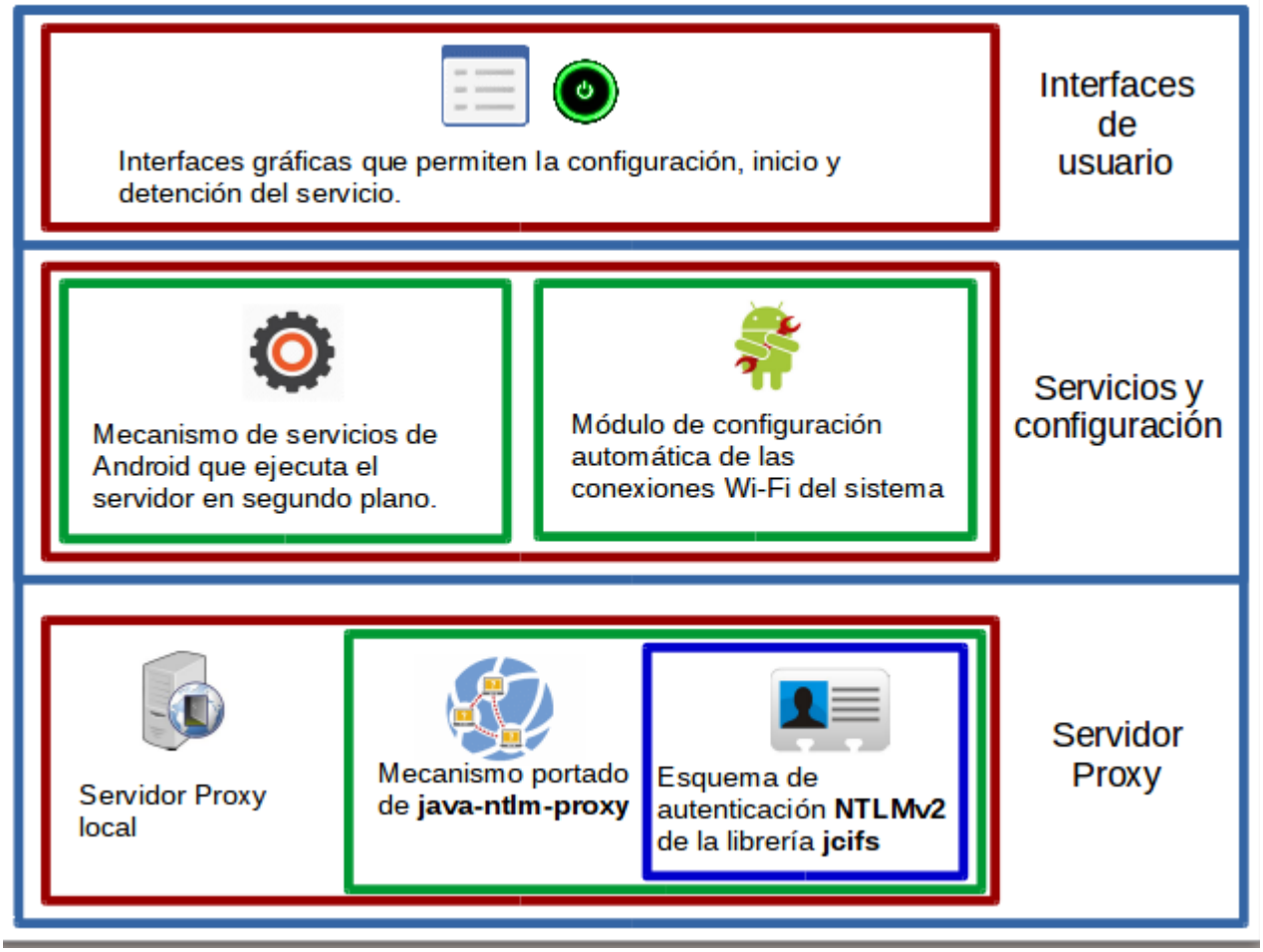

Ilustración 3: Esquema de propuesta

2.2.Desglose de responsabilidades

Como parte de la metodología OpenUP se encuentra la definición de responsabilidades para cada participante en el proyecto, definiéndose en los diferentes roles a desempeñar, como se muestra en la Tabla 2.

Rol	Responsabilidad	Nombre
Analista	Representa al cliente y las preocupaciones de los usuarios finales mediante la recopilación de información a los interesados, para entender el problema a resolver, por la captura y fijación de prioridades para las necesidades.	Laura Pérez Vera Jorge A. Díaz Orozco
Arquitecto	Responsable de definir la arquitectura de *software*, que incluye tomar las principales decisiones técnicas que limitan el diseño y la implementación del sistema.	Laura Pérez Vera Jorge A. Díaz Orozco
Programador	Responsable del desarrollo de una parte del sistema, incluyendo el diseño que se ajuste a la arquitectura, de prototipos de la interfaz de usuario y luego la aplicación, la unidad de pruebas y la integración de los componentes que forman parte de la solución.	Laura Pérez Vera Jorge A. Díaz Orozco
Líder de proyecto	Responsable de la planificación del proyecto, coordina las interacciones con las partes	Laura Pérez Vera Jorge A. Díaz Orozco

	interesadas y mantiene al equipo de proyecto centrados en el cumplimiento de los objetivos del mismo.	
Stakeholder	Representa los grupos de interés cuyas necesidades deben ser satisfechas por el proyecto.	MsC. Allan Pierra Fuentes Ing. Dairelys García Rivas
Probador	Responsable de las actividades básicas de la prueba de esfuerzo. Esas actividades incluyen la identificación, definición, implementación y realización de las pruebas necesarias, así como registrar los resultados de las pruebas y análisis de los resultados.	Laura Pérez Vera Jorge A. Díaz Orozco

Tabla 2: Tabla de Responsabilidades

2.3.Listado de requisitos funcionales y no funcionales

La administración de requisitos es una de las actividades iniciales de OpenUP, para la posterior confección de la especificación de requisitos, donde se definen, de manera priorizada, los requisitos funcionales y no funcionales de la aplicación. La especificación de requisitos forma parte de los artefactos definidos en la metodología OpenUP, del que se toma la siguiente lista de requisitos. La misma cuenta con 21 requisitos funcionales, de ellos 18 son de prioridad alta, 2 de prioridad media y 1 de prioridad baja.

Requisitos funcionales (RF)

Un **requisito funcional** define servicios o funciones que el cliente requiere de un sistema (Ian Sommerville 2005).

Prioridad Alta

1. Proporcionar una interfaz para que el usuario introduzca su nombre de usuario y contraseña.

2. Habilitar la opción para que el usuario pueda iniciar el servicio.

3. Habilitar la opción para que el usuario pueda detener el servicio.

4. Permitir al usuario especificar el dominio al que pertenece para realizar la autenticación.

5. Permitir al usuario especificar la dirección y puerto del servidor *proxy* al que se conectará.

6. Permitir al usuario especificar el puerto local por el que funcionará el servicio.

7. Permitir al usuario especificar la lista de direcciones para las cuales no desea utilizar *proxy*.

8. Almacenar las credenciales y configuración establecidas por el usuario para futuras ejecuciones.

9. Iniciar el servicio desde el formulario de configuración.

10. Detener el servicio desde el propio formulario de configuración.

11. Iniciar el servicio desde una interfaz (*widget*) en el escritorio del dispositivo.

12. Detener el servicio desde una interfaz (*widget*) en el escritorio del dispositivo.

13. Configurar la *wifi* para excluir del proceso de autenticación las direcciones especificadas por el usuario.

14. La solución deberá integrarse con las configuraciones generales del SO donde sea ejecutada.

15. La solución deberá configurar la conexión *wifi* activa para utilizar el servicio una vez iniciado el mismo.

16. Restablecer la configuración de la *wifi* activa a los valores por defecto una vez detenido el servicio.

17. Configurar la *wifi* del dispositivo una vez que el mismo se encuentre conectado, acorde al estado del servicio.

18. No requerir permisos de administración.

Prioridad Media

1. Separar los datos de autenticación de las opciones avanzadas, mostrando inicialmente sólo los primeros.

2. Brindar una opción para mostrar y ocultar las opciones avanzadas.

Prioridad Baja

1. Permitir al usuario mostrar su contraseña para comprobar que no contiene errores.

Requisitos no funcionales (RNF)

Software

1. La solución deberá ser funcional para el SO Android a partir de la versión 3.1.

2. La solución para el cumplimiento de su objetivo deberá estar previamente instalada en un dispositivo con navegación mediante un servidor *proxy* con autenticación NTLM y/o NTLMv2.

Confiabilidad

1. Los datos del usuario se enviarán de forma segura al servidor *proxy*, cifrando la contraseña para su almacenamiento con un algoritmo seguro utilizando como sal[12] el número serial del dispositivo.

Restricciones de diseño

1. El sistema no deberá almacenar contraseñas en texto plano.

12 Comprende bits aleatorios que se usan como una de las entradas en una función derivadora de claves. La otra entrada es habitualmente una contraseña. La salida de la función derivadora de claves se almacena como la versión cifrada de la contraseña.

2. La solución contará con soporte de lenguaje en español, inglés y francés dependiendo del dispositivo donde sea instalada.

3. La configuración estándar de la solución deberá corresponder con la configuración de la UCI.

Soporte

1. La solución se implementará utilizando Java, con Android SDK 18.

2. Utilizar como IDE de desarrollo Android *Studio* rc 1.0.

2.4.Análisis del dominio

Como parte de la fase de elaboración vista en el epígrafe 1.3.1 de la metodología OpenUP, se realiza un análisis del dominio y definición de la arquitectura del sistema que brinda al equipo de proyecto un prototipo funcional inicial de la solución, este se irá modificando según los requisitos que busque el cliente, hasta llegar a la propuesta final.

2.4.1. Modelo de dominio

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real, no de componentes de *software* (Craig Larman 2003).

La Ilustración 4 muestra el modelo de dominio realizado, teniendo en cuenta lo planteado en la descripción del problema.

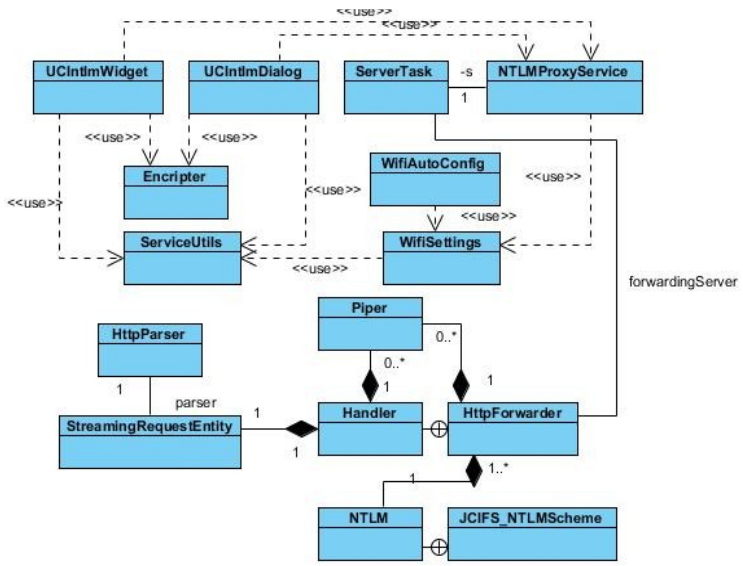

Ilustración 4: Modelo de dominio

2.4.2. Diagrama de clases (Ver Anexo 2)

Como se ve en el Anexo 2, las clases se encuentran ordenadas en paquetes por los servicios que brindan, estos son: el paquete **u**i (va referido a las interfaces con las que interactúa el usuario que son el formulario y el *widget*), el paquete **service** (presenta lo relacionado con los servicios que brinda la aplicación, entre ellos la configuración de la *wifi* de forma automática y el trabajo con el *proxy* NTLM) y el paquete **core** (representa la interacción con el SO). Estos paquetes además presentan comunicación basada en una abstracción que proporciona bajo acoplamiento entre capas. Por lo tanto, se afirma que el estilo arquitectónico usado es **N-Capas.**

2.5.Patrones utilizados en el desarrollo de la solución

Los desarrolladores orientados a objetos con experiencia (y otros desarrolladores de *software*) acumulan un repertorio tanto de principios generales como de soluciones basadas en aplicar ciertos estilos que les guían en la creación de *software*. Estos principios y estilos, si se codifican con un formato estructurado

que describa el problema y la solución y se les da un nombre, podrían llamarse **patrones** (Craig Larman 2003).

A continuación se especifican los patrones que serán utilizados en la solución a desarrollar.

2.5.1. Patrones GoF

Los patrones “Banda de los Cuatro” (GoF, por sus iniciales en inglés *Gang of Four*), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Existen tres tipos de patrones: los de **creación** que abstraen el proceso de creación de instancias, **estructurales** que se ocupan de cómo las clases y objetos son utilizados para componer estructuras de mayor tamaño y de **comportamiento** que se refieren a los algoritmos y a la asignación de responsabilidades entre objetos (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 1995).

A continuación se identifican los patrones utilizados en el desarrollo de la aplicación, de acuerdo a su clasificación.

Creación

***Abstract Factory*:** Dado un conjunto de clases abstractas relacionadas, el patrón *Abstract Factory* permite el modo de crear instancias de estas clases abstractas desde el correspondiente conjunto de subclases concretas. Proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta. El patrón *Abstract Factory* puede ser muy útil para permitir a un programa trabajar con una variedad compleja de entidades externas, tales como diferentes sistemas de ventanas con una funcionalidad similar (Francisco Javier Martínez Juan 2000).

En la solución planteada este patrón es utilizado con el objeto **Intent**, permitiendo cuando se instancia especificar el tipo de clase que va a crear. La Ilustración 5 especifica el uso del patrón en el desarrollo de la solución.

```
Intent intent = new Intent(this, NTLMProxyService.class);

if (domain.getText().toString().equals("")
        || server.getText().toString().equals("")
        || inputport.getText().toString().equals("")
        || outputport.getText().toString().equals("")) {

    Toast.makeText(getApplicationContext(), "There is no enough data to start the service",
            Toast.LENGTH_SHORT).show();
    startButton.setChecked(false);
    return;

}

if (!ServiceUtils.isMyServiceRunning(this)) {
    intent.putExtra("user", user.getText().toString());
    intent.putExtra("pass", pass.getText().toString());
    intent.putExtra("domain", domain.getText().toString());
    intent.putExtra("server", server.getText().toString());
    intent.putExtra("inputport", inputport.getText().toString());
    intent.putExtra("outputport", outputport.getText().toString());
    intent.putExtra("bypass", bypass.getText().toString());

    startService(intent);
    UCIntlmWidget.actualizarWidget(this.getApplicationContext(),
            AppWidgetManager.getInstance(this.getApplicationContext()),
            "on");
    disableAll();
    saveConf();
} else {
    stopService(intent);
    enableAll();
    UCIntlmWidget.actualizarWidget(this.getApplicationContext(),
            AppWidgetManager.getInstance(this.getApplicationContext()),
            "off");
}
```

Ilustración 5: Patrón *Abstract Factory*

Comportamiento

Observador *(Observer)*: Permite a los objetos captar dinámicamente las dependencias entre objetos, de tal forma que un objeto notificará a los objetos dependientes de él cuando cambia su estado, siendo actualizados automáticamente (Francisco Javier Martínez Juan 2000).

En la solución propuesta, este patrón es utilizado al crear clases que extiendan de la clase nativa de Android **BroadcastReceiver**, esta clase cumple con la principal funcionalidad de esperar eventos para su

gestión. La Ilustración 6 especifica la clase **WifiAutoConfig** que extiende de **BroadcastReceiver**, para esperar el cambio de estado de la *wifi* del dispositivo.

```
/**
 * Esta clase configura automáticamente la WiFi a la que se conecte el sistema
 * Espera en segundo plano la ocurrencia del evento android.net.wifi.STATE_CHANGE
 */
public class WifiAutoConfig extends BroadcastReceiver
```

Ilustración 6: Patrón *Observer*

2.5.2. Patrones Generales de *Software* para la Asignación de Responsabilidades

Patrones de Asignación de Responsabilidad (GRASP, por sus siglas en inglés *General Responsibility Assignment Software Patterns*) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones segun

- **Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. La solución planteada se desarrolla bajo el concepto de 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 (Craig Larman 2003). En el diagrama de clases (ver Ilustración 14) se puede ver la aplicación de este patrón, por ejemplo la clase **ServiceUtils**, es la encargada de demostrar si el servicio se mantiene activo, sólo ella puede responder a esa funcionalidad. Además la clase **WifiSettings** es la encargada de modificar los parámetros de la *wifi* del dispositivo.
- **Creador:** Ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases (Craig Larman 2003). En el desarrollo de la solución planteada se trabaja con clases que realizan esta función, como es la clase **NTLM** del paquete ntlmv2 que necesita de instancias de la clase **HttpForwarder**, sin embargo ella misma realiza su instanciación que luego la utiliza NTLM.

- **Alta Cohesión:** Es la meta principal que se tiene en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Siguiendo con este principio, el diseño de clases se encuentra construido teniendo en cuenta que una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo que hacer (Craig Larman 2003). En el diagrama de clases de la solución que se muestra en la Ilustración 14, la clase **Encripter** presenta sólo dependencias con las clases que se encuentran en el paquete **ui** que representa las interfaces. Esta clase, a pesar de tener gran responsabilidad de seguridad, tiene dependencias con las clases necesarias.

- **Bajo acoplamiento:** El bajo acoplamiento es un principio que se debe tener en cuenta durante las decisiones de diseño. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más re-utilizables, que acrecienten la oportunidad de una mayor productividad. En el diseño de clases de la solución propuesta las clases se encuentran lo menos ligadas entre sí que se pueda de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la re-utilización y disminuyendo la dependencia entre las clases (Craig Larman 2003). El paquete **service** (ver Ilustración 14) se encarga de manejar el servicio *proxy*, está diseñado sobre la idea de si se quisiera añadir un nuevo servicio al sistema la tarea se convertiría en algo sencillo, añadiendo una nueva clase o mediante su modificación.

2.6.Descripción de los actores del sistema

En el Lenguaje Unificado de Modelado (UML), un **actor** "especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto" (*Unified Modeling Language (UML)* 2014).

La descripción del actor que interactúa con el sistema que corresponde a la solución propuesta, se muestra a continuación:

Actores	Justificación
Usuario	Es la persona que inicia el servicio y lo detiene para el trabajo con la navegación a Internet.

Tabla 3: Descripción de los actores del sistema

2.7.Diagrama de casos de uso del sistema

En UML, un **diagrama de casos de uso** es una forma de diagrama de comportamiento UML, la descripción escrita del comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio (*Unified Modeling Language (UML)* 2014).

Siguiendo lo anteriormente planteado, se muestra la Ilustración 7.

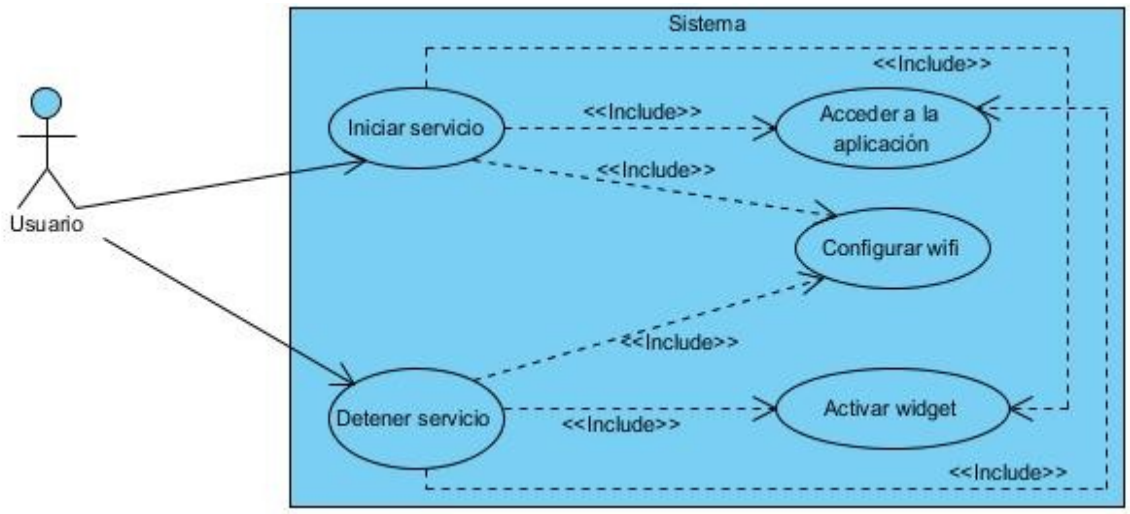

Ilustración 7: Diagrama de casos de uso del sistema

2.7.1. Patrones de casos de uso utilizados

En la elaboración de los casos de uso se utilizan diferentes patrones que permiten establecer los diferentes comportamientos que deben existir en el sistema. La aplicación de estos patrones de casos de uso son mejores prácticas para su modelado (Craig Larman 2003).

1. **Inclusión:** Se incluye una relación del caso de uso base al caso de uso de inclusión. La Ilustración 7 muestra el trabajo con este patrón, tomando como casos de uso base Iniciar servicio y Detener servicio.

2.8.Descripción de los casos de uso del sistema

A continuación se realiza la descripción de los casos de uso de la solución.

Caso de Uso	Iniciar Servicio	
Objetivo	Iniciar el servicio para la navegación por Internet mediante el dispositivo inteligente.	
Actor	Usuario	
Resumen	El caso de uso inicia cuando el Actor selecciona Iniciar Servicio. La aplicación hará las configuraciones necesarias para permitir la navegación a Internet.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	Debe estar conectado a una red *wifi* institucional.	
Postcondiciones	El usuario del dominio habrá logrado la navegación por Internet.	
Flujo de eventos		
Flujo básico Normal de Eventos		
	Actor	**Sistema**
1	Iniciar la aplicación.	
2	Introducir los datos de dominio.	
3	Seleccionar Opciones Avanzadas.	
4	Introducir los datos de configuración del servidor *proxy*.	
5	Seleccionar Iniciar Servicio.	
6	6	Iniciar Servicio con los datos introducidos por el Actor.
7		Terminar Caso de Uso.

Caso de Uso	Detener Servicio
Objetivo	Detener el servicio cuando el usuario no desee navegar por Internet desde su dispositivo inteligente.
Actor	Usuario
Resumen	El caso de uso inicia cuando el Actor selecciona Detener Servi-

	cio. La aplicación hará las configuraciones necesarias para detener el servicio.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Debe estar iniciado el servicio.
Postcondiciones	El servicio se detiene y la *wifi* se vuelve a configurar para trabajar sin *proxy*.
Flujo de eventos	
Flujo básico Normal de Eventos	

	Actor	**Sistema**
1	Seleccionar la opción Detener Servicio.	
2		Detener el servicio.
3		Configurar la *wifi*.
4		Terminar Caso de Uso.

Caso de Uso	Configurar *wifi*
Objetivo	Configurar la *wifi* para el trabajo con el *proxy*
Actor	Usuario
Resumen	Cuando el usuario inicia el servicio la *wifi* deberá configurarse automáticamente con los valores entrados por el usuario. Si el usuario detiene la misma volverá a configurarse.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Debe estar conectado a la *wifi*
Postcondiciones	La configuración de la *wifi* quedará establecida dependiendo si el usuario inicia o no el servicio
Flujo de eventos	
Flujo básico Normal de Eventos	

	Actor	**Sistema**
1	Iniciar Servicio	
2		Cambiar el nombre de Host de *proxy* a 127.0.0.1 y los campos **puerto *proxy*** y **omitir *proxy*** según los parámetros que haya configurado el Actor en la configuración de la *wifi* activa.
3		Terminar Caso de Uso.

Caso de Uso	Acceder a la aplicación	
Objetivo	El Usuario puede acceder al formulario principal de la aplicación.	
Actor	Usuario	
Resumen	El caso de uso inicia cuando el Actor selecciona el icono de la aplicación.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	La aplicación debe estar previamente instalada en el dispositivo.	
Postcondiciones	El Actor accede al formulario.	
Flujo de eventos		
Flujo básico Normal de Eventos		
	Actor	**Sistema**
1	Instalar la aplicación.	
2	Seleccionar la aplicación en el lanzador.	
3		Acceder al formulario de autenticación.
4		Terminar Caso de Uso

Caso de Uso	Activar *widget*	
Objetivo	El Usuario puede iniciar o detener el servicio desde el *widget.*	
Actor	Usuario	
Resumen	El caso de uso inicia cuando el Actor selecciona el *widget* para iniciar o detener el servicio.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	El Actor debe haber introducido los datos en el formulario principal al menos una vez y haber posicionado el widget en el escritorio del dispositivo.	
Postcondiciones	El Actor inicia o detiene el servicio desde el widget.	
Flujo de eventos		
Flujo básico Normal de Eventos		
	Actor	**Sistema**
1	Acceder al widget de la aplicación.	
2		Si el servicio se encuentra detenido, iniciar, sino lo detiene.
3		Muestra el texto informando al Actor del estado del servicio.
4		Cambia el color del icono dependiendo del estado.

5		Terminar Caso de Uso.

2.9.Conclusiones parciales

En este capítulo se identificó una visión general de lo que será la solución del sistema, utilizando para ello como metodología de desarrollo OpenUP. La propuesta de solución presentada funciona como *proxy* de autenticación, sirviendo de intermediario entre el cliente que envía la solicitud y el *proxy*. Este intermediario se ocupará del mecanismo de autenticación en lugar del cliente. Se precisaron un total 21 requisitos funcionales, de ellos 18 son de prioridad alta, 2 de prioridad media y 1 de prioridad baja, desglosados en 5 casos de uso. Además se representó la lógica del sistema mediante el diagrama de clases, el modelo del dominio y el diagrama de casos de uso.

Capítulo 3. Implementación y realización de pruebas.

En este capítulo se abordará la fase de implementación y prueba de la solución propuesta, guiada por el análisis y diseño de la misma, abordados en el Capítulo 2. Análisis y diseño de la solución propuesta. Una vez finalizada la fase de implementación se realizará el diseño y estudio de los diferentes casos de prueba a aplicar a la solución con el fin de asegurar la calidad y funcionamiento de la misma.

3.1. Implementación

La implementación de la solución desarrollada se corresponde con la fase de construcción definida por la metodología OpenUP. Para la obtención del resultado esperado fueron necesarias 3 iteraciones, el resultado de cada iteración y las modificaciones realizadas en cada una de ellas se detalla a continuación.

Iteración	Descripción	Duración
1	- Aplicación funcional que permite la conexión a Internet sólo desde dominio UCI. - *Widget* de la aplicación.	4 semanas
2	- Se agregan los campos de configuración avanzada para su utilización de forma externa a la UCI. - Se añade la configuración que permite guardar los datos ya entrados en el sistema. - Se agrega el funcionamiento en segundo plano de la aplicación. - Se realiza mejora al diseño. - Se agrega un *scroll* para que el contenido del formulario pueda ser visualizado en pantallas de 2.7". - Se agrega la traducción en Español de la aplicación.	4 semanas
3	- Mejoras al diseño. - Se agrega la opción de mostrar contraseña. - Permite la configuración de la *wifi* a la que esté conectada el dispositivo. - Establece compatibilidad con dispositivos que utilizan la API 8. - Se retira el soporte para dispositivos con sistema inferior a Android 3.1.	4 semanas

Tabla 4: Modificaciones en fase de implementación

3.1.1. Estándar de codificación

Una de las buenas prácticas utilizadas en la construcción de la solución planteada lo constituye la adopción del estándar de codificación ***CamelCase***, asegurando que el código exprese claramente el propósito del mismo y agilice el proceso de refactorización, que sea legible, entendible y refleje un estilo armonioso.

3.1.2. Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de *software* es dividido en componentes y muestra las dependencias entre estos. Los diagramas de componentes prevalecen en el campo de la arquitectura de *software* pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestran la organización y las dependencias entre un conjunto de componentes (Roger S. Pressman 2005).

El Diagrama de Componentes de la solución propuesta se representa en la Ilustración 8.

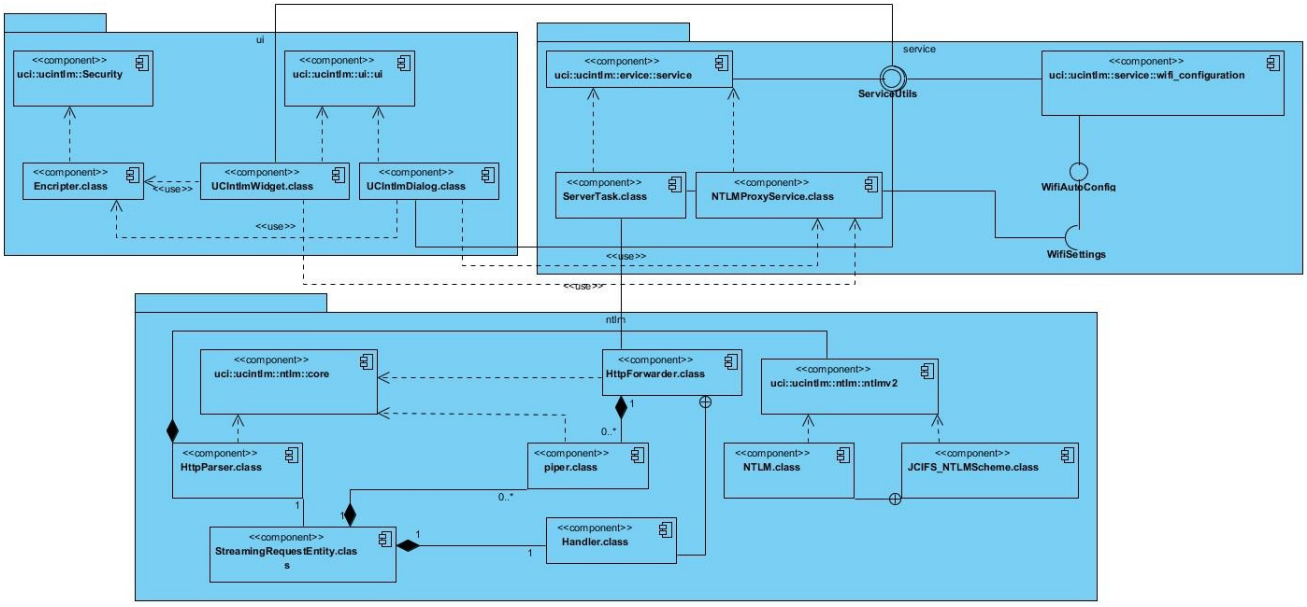

Ilustración 8: Diagrama de Componentes

3.1.3. Diagrama de despliegue

Es un tipo de diagrama del UML que se utiliza para modelar la disposición física de los artefactos del *software* en nodos (*Unified Modeling Language (UML)* 2014).

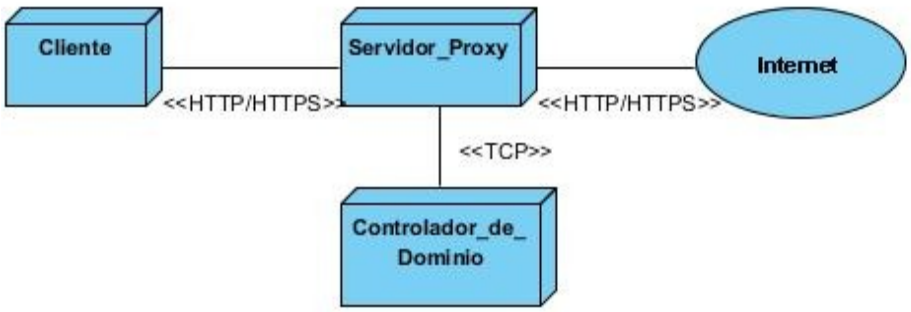

Ilustración 9: Diagrama de Despliegue

3.2. Pruebas de s*oftware*

Durante y después del proceso de implementación, el programa que se está desarrollando debe ser comprobado para asegurar que satisface los requerimientos del cliente (Ian Sommerville 2005). Los autores del presente trabajo de investigación siguiendo lo antes expuesto, hicieron uso de las **técnicas de evaluación estáticas** en las fases de análisis y diseño de la solución. Se realizaron en forma de **revisiones** de carácter informal a los diferentes artefactos que fueron necesarios generar para la posterior implementación. Por otra parte, la metodología OpenUP define las pruebas funcionales basadas en casos de prueba, razón por la cual una vez finalizadas las diferentes fases de implementación se llevaron a cabo **técnicas de evaluación dinámicas**, siguiendo el **método de caja negra** en la generación de casos de prueba. Los tipos de prueba que se realizaron en esta fase fueron **tipos de prueba de funcionalidad**, sirviendo este tipo de prueba para validar si el comportamiento observado del *software* cumple o no con sus especificaciones, definiéndose además como nivel de prueba las **pruebas de aceptación**.

3.2.1. Pruebas de aceptación

Cuando se realizan este tipo de pruebas, el producto está listo para implantarse en el entorno del cliente. Están enfocadas a probar los requisitos y los criterios de aceptación, si no se consigue demostrar el

incumplimiento de tales requisitos el cliente deberá aceptar el producto. En caso de un producto de interés general, se realizan pruebas con varios usuarios que reportarán sus valoraciones sobre el producto. Para la generación de casos de prueba de aceptación se utilizan técnicas de caja negra (Natalia Juristo, Ana M. Moreno, Sira Vegas 2005).

3.2.1.1. Casos de pruebas de aceptación

En correspondencia con los casos de usos más importantes, se diseñaron los casos de prueba que son basados en escenarios, como se observa en la Tabla 5.

Id del Escenario	Escenario	Descripción	Respuesta del Sistema	Resultado de la Prueba
EC1	Escenario 1: Iniciar Servicio de forma exitosa.	El usuario inicia el servicio ya sea desde el *widget* o desde el formulario, permitiendo navegar por Internet desde su dispositivo	El Sistema muestra al Usuario que la aplicación ya se encuentra en ejecución	Satisfactoria
EC2	Escenario 2: Campos Vacíos	El usuario cuando accede al formulario principal inicia el servicio con los campos de usuario, contraseña y las direcciones que no necesitan *proxy* vacíos	El Sistema muestra que la aplicación se encuentra en ejecución pero no establece el acceso a Internet	Satisfactoria
EC3	Escenario 3: Usuario sin permiso de administrador	El usuario inicia servicio en un dispositivo donde no tiene permiso de administrador	El Sistema inicia de forma exitosa el servicio	Satisfactoria
EC4	Escenario 4: Cambio de	El usuario inicia servicio en un dispositivo donde su	El Sistema inicia de forma exitosa el servicio	Satisfactoria

	configuración	dominio y configuración no son los de la UCI		
EC5	Escenario 5: Campos vacíos de Dominio, Servidor, Puerto de Entrada y Puerto de Salida	El usuario cuando accede al formulario principal inicia el servicio con los campos de Dominio, Servidor, Puerto de Entrada y Puerto de Salida vacíos	El Sistema muestra un mensaje de texto notificando al usuario de que el servicio no puede ser iniciado	Satisfactoria
EC6	Escenario 6: Detener Servicio de forma exitosa	El usuario detiene el servicio ya sea desde el *widget* o desde el formulario	El Sistema muestra al Usuario que el servicio se ha detenido	Satisfactoria

Tabla 5: Escenarios de Prueba

3.2.1.2. Resultados de las pruebas de aceptación

Las pruebas de Aceptación fueron realizadas al final de cada iteración, teniendo en cuenta que cada iteración termina con un prototipo funcional de la solución. Al finalizar la primera iteración se obtuvo como resultado un total 7 No Conformidades (NC), de ellas 3 fueron clasificadas de Significativas (S), 2 de No Significativas (NS) y 2 Recomendaciones (R). En la segunda iteración finalizó con 4 NC, de ellas 2 de tipo S, 1 de tipo NS y 1 de tipo R. En la iteración final no se obtuvo ninguna NC. La explicación anterior se puede ver reflejada en la Ilustración 10.

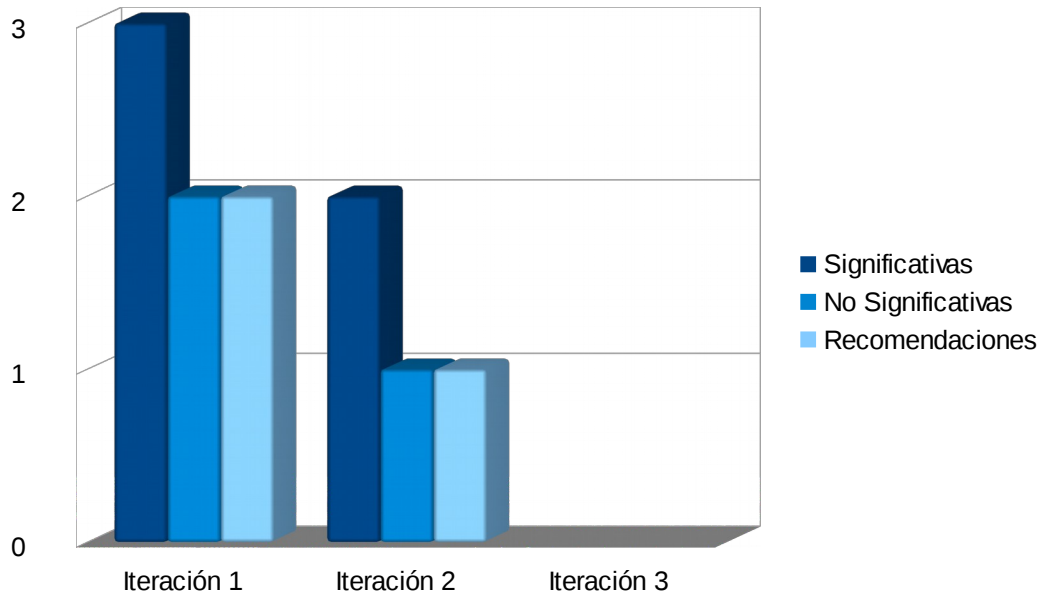

Ilustración 10: Resultado de las pruebas de Aceptación

NC de carácter S detectadas en la primera iteración

- El EC1 no se cumple al no permitir que las aplicaciones accedan a Internet.
- El EC4 no se cumple satisfactoriamente al no permitir trabajar con otras configuraciones de *proxy* que no sea de la UCI.
- El EC5 no se cumple al no permitir que las aplicaciones accedan a Internet dejando los campos especificados vacíos.

NC de carácter S detectadas en la segunda iteración

- El EC6 no se cumple de forma satisfactoria al no permitir detener el servicio desde el *widget*.
- El EC1 no se cumple de forma satisfactoria al no permitir iniciar el servicio desde el *widget*.

3.3. Resultados del uso de UCIntlm

La aplicación UCIntlm sigue el principio de código abierto, lo que significa que su código fuente se encuentra a disposición de los usuarios, con el objetivo de que la comunidad pueda sugerir modificaciones, encontrar errores, entre otras posibilidades. Los autores del presente trabajo de investigación utilizaron el *blog* HumanOS[13] (http://humanos.uci.cu) para la publicación de las versiones y las mejoras hechas hasta llegar al producto final. La siguiente imagen muestra la cantidad de descargas que tuvieron las diferentes versiones de la aplicación en HumanOS, demostrando el interés de la comunidad en la misma.

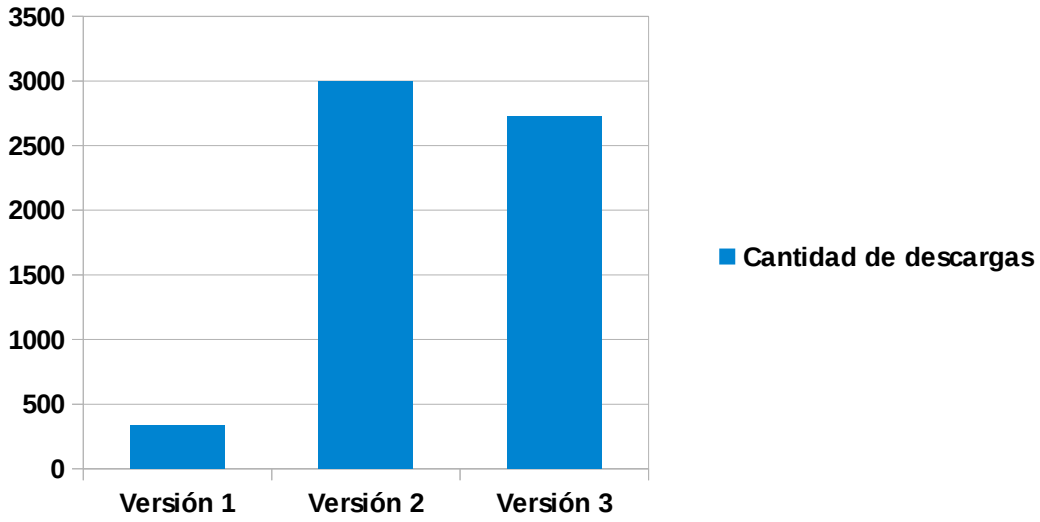

Ilustración 11: Estadística de descargas

Una vez finalizada la última versión de la aplicación, se decide realizar una encuesta a la comunidad (Ver Anexo 1) arrojando diferentes resultados. La Ilustración 12 muestra las respuestas obtenidas en cuanto a la frecuencia con la que los usuarios utilizan la aplicación, de un total de 207 usuarios que respondieron esta pregunta, se obtiene que en su mayoría es usada una o más veces por semana, demostrando la utilidad de la misma.

13 *Blog* que busca mantener a la comunidad de usuarios de la UCI informada sobre Linux.

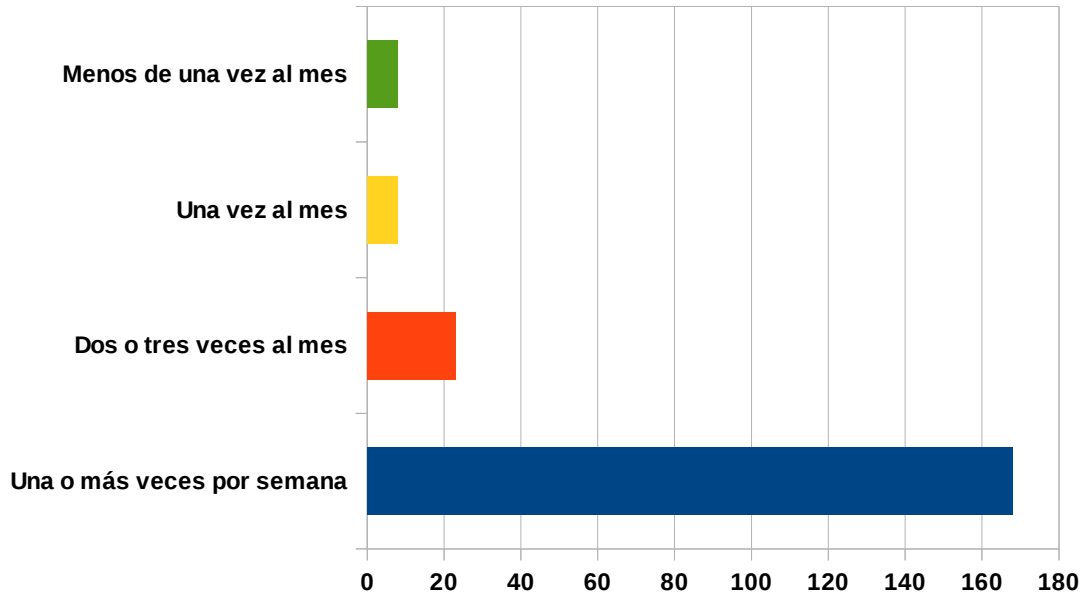

Ilustración 12: Frecuencia de uso

La Ilustración 13 muestra los resultados obtenidos al preguntar a la comunidad que tuviera conocimiento de otras aplicaciones en Android que intentaran cumplir la misma funcionalidad, como clasificarían UCIntlm. Los resultados demuestran que de un total de 196 usuarios que respondieron esa pregunta, 119 la clasificaron como Mejor 39 como Igual y solo 9 como Peor.

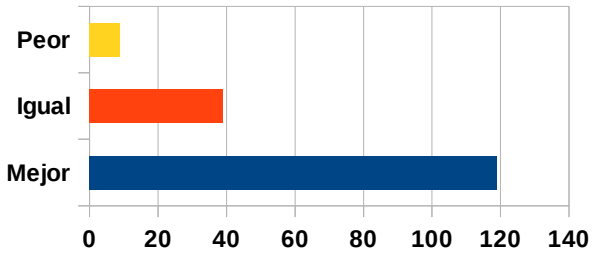

Ilustración 13: Comparación en la encuesta

3.4. Conclusiones parciales

En este capítulo se definieron las 3 iteraciones que fue necesario realizar hasta llegar a la solución óptima. En las pruebas de Aceptación en total se detectaron 11 NC, de ellas 5 de tipo S, 3 de tipo NS y 3 de tipo R, siendo todas resueltas. Esto garantiza que la versión implementada en la iteración final cumpla con las necesidades del cliente, garantizando así la calidad y efectividad. Además se pudo verificar la aceptación que tiene la aplicación en la comunidad según los resultados obtenidos en la encuesta realizada.

Conclusiones

Con la culminación del presente trabajo de diploma se cumplieron cada uno de los objetivos trazados, teniendo de manera general los siguientes aspectos:

- A partir del estudio realizado en la etapa inicial de la investigación se puede afirmar que las aplicaciones Autoproxy, Sandroproxy y Proxydroid no se ajustan a las necesidades requeridas.
- El estudio de las diferentes herramientas y tecnologías permitieron seleccionar OpenUP como metodología de desarrollo, Android *Studio* rc1.0 como IDE, *Visual Paradigm* como herramienta CASE, Java como lenguaje de programación y Android SDK como *kit* de desarrollo. La aplicación Java-NTLM-Proxy v0.2 y la librería JCIFS v1.3.17 fueron utilizadas como parte de la solución.
- Las pruebas realizadas a la aplicación permiten afirmar que UCIntlm establece la conexión a Internet con autenticación NTLM y NTLMv2 de las aplicaciones en los teléfonos inteligentes con SO Android.

Recomendaciones

Para garantizar el correcto funcionamiento de UCIntlm se realizan las siguientes recomendaciones para su futuro desarrollo:

- Adicionar el soporte para otros métodos de autenticación entre ellos *Kerberos*.
- Establecer como configuración que se comporte como parte de las preferencias del sistema.
- Especificar que en terminales con Android v4.4 es necesario que no se incluyan caracteres especiales en la configuración de **omitir *proxy* para**.

Referencias Bibliográficas

ANDRES TARASCO ACUÑA, 2008, *Defeating Win32 Network Security with NTLM*. 2008.

Android Studio Overview | Android Developers, 2014. [online], [Accessed 24 February 2015]. Available from: http://developer.android.com/tools/studio/index.html

Autoproxy, 2013. *Googleplay* [online], Available from: play.google.com

BYOUS,JON, 2005, *Java technology: The early years. Sun Developer Network*. de abril de 2005.

CARLOS AGUILAR RODRÍGUEZ, 2010, *ANALISIS DE LAS VENTAJAS DE USAR UN SERVIDOR PROXY EN UNA DEPENDENCIA DE GOBIERNO MUNICIPAL*.

CRAIG LARMAN, 2003, *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2da. Prentice Hall.

DANIEL GONZÁLEZ, MIGUEL HERNÁNDEZ and JOSÉ GARDUÑO, 2003, *JAVA para estudiantes*. 3. México.

ERIC GLASS, 2006, *The NTLM Authentication Protocol and Security Support Provider*.

ERICH GAMMA, RICHARD HELM, RALPH JOHNSON and JOHN VLISSIDES, 1995, *Desing Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley.

FRANCISCO JAVIER MARTÍNEZ JUAN, 2000, *GUÍA DE CONSTRUCCIÓN DE SOFTWARE EN JAVA CON PATRONES DE DISEÑO*. Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo : Universidad de Oviedo.

GREG FERRO, 2014, *Servidores proxy y servidores de proxy inversos*. June 2014.

HOSSEIN FALAKI, DIMITRIOS LYMBEROPOULOS, RATUL MAHAJAN, SRIKANTH KANDULA and DEBORAH ESTRIN, 2010, A first look at traffic on smartphones. . 2010.

IAN SOMMERVILLE, 2005, *Ingeniería de Software*. 5ta.

Java NTLM Proxy, 2007. [online], Available from: http://sourceforge.net/projects/java-ntlm-proxy/

JENNIFER G. STEINER, CLIFFORD NEUMAN and JEFFREY I. SCHILLER, 2012, *Kerberos Overview- An Authentication Service for Open Network Systems*.

KEN HYERS, 2014, Strategy Analytics: Global Smartphone Shipments. [online]. 2014. Available from: www.strategicanalytics.com

NATALIA JURISTO, ANA M. MORENO and SIRA VEGAS, 2005, *TÉCNICAS DE EVALUACIÓN DE SOFTWARE*. 17 October 2005.

OpenUP, 2012. *OpenUP* [online], Available from: http://epf.eclipse.org

PATRICIA LÓPEZ, 2014, *Herramienta CASE Visual Paradigm*. Universidad Cantabria.

ProxyDroid, 2014. *Googleplay* [online], Available from: play.google.com

ROBERT W. SEBESTA, 2012, *CONCEPTS OF PROGRAMMING LANGUAGES.* 10. New Jersey.

ROGER S. PRESSMAN, 2005, *Ingeniería del Software*. 6. México.

SandroProxy, 2014. *Googleplay* [online], Available from: play.google.com

SHENG LIANG, 2000, *Java Native Interface: Programmer's Guide and Specification*.

SONIA PINZÓN and JUAN CARLOS GUEVARA BOLAÑOS, 2006, *La gestión, los procesos y las metodologías de desarrollo de software*. June 2006.

The Java CIFS Client Library, 2008. *JCIFS* [online], Available from: https://jcifs.samba.org/

Unified Modeling Language (UML), 2014. [online], Available from: UML.html

What is Android?: Android Runtime, 2006. *Developers* [online], Available from: http://developer.android.com

Glosario de siglas y términos

• Android: sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, *tablets* y también para relojes inteligentes, televisores y automóviles.

• Código Abierto: término con el que se conoce al software distribuido y desarrollado libremente.

• Código Fuente: término referido al conjunto de líneas de texto que se corresponden con las instrucciones que debe seguir la computadora para ejecutar algún programa, estableciendo que en el código fuente de un programa está escrito por completo su funcionamiento.

• *Hardware*: corresponde a todas las partes tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.

• *Host* : (en español, anfitrión), término utilizado para referirse a las computadoras conectadas a una red, que proveen y utilizan servicios de ella.

• HTTP (*HyperText Transfer Protocol*): (en español, Protocolo de Transferencia de Hipertexto), es un método de intercambio de información en la World Wide Web mediante el cual se transfiere el contenido de las páginas web a una computadora.

• *Software*: programas del ordenador. Se refiere a instrucciones que se ejecutan por medio de un equipo de cómputo.

• *Software* Libre: denominación del *software* que respeta la libertad de todos los usuarios que adquirieron el producto y por tanto, una vez obtenido el mismo, puede ser usado, copiado, estudiado, modificado, y redistribuido libremente de varias formas.

Anexos

Anexo 1.Encuesta aplicada

La siguiente encuesta busca obtener información del funcionamiento de la aplicación UCIntlm y las facilidades que le reporta a usted como usuario el uso en su dispositivo inteligente.

1. ¿Cuánto tiempo lleva utilizando la aplicación?

__ Menos de un mes
__ Entre uno y seis meses
__ Más de seis meses

2. ¿Con qué frecuencia utiliza la aplicación?

__ Una o más veces a la semana

__ Dos o tres veces al mes

__ Una vez al mes

__ Menos de una vez al mes

3. ¿Cuál es su grado de satisfacción general con la utilización de la aplicación?

__Completamente satisfecho

__Satisfecho

__Insatisfecho

__Completamente insatisfecho

4. En comparación con otras aplicaciones que usted conozca que sean similares en funcionalidad a UCIntlm ¿cómo clasificaría a esta última?

__Mucho mejor

__Algo mejor

__Más o menos igual

__Algo peor

__Mucho peor

__No lo sé

5. ¿Usted como usuario ha recomendado el uso de esta aplicación a otras personas?

__Sí __No

6. ¿Qué porciento de las aplicaciones que utiliza en su dispositivo necesitan de tener previamente instalado UCIntlm?

__20% __50% __Más del 50%

7. ¿En qué dispositivo inteligente usted utiliza UCIntlm?

__ Teléfono inteligente __Tableta

8. ¿Ha tenido usted algún problema a la hora de usar UCIntlm?

__Sí __No

9. ¿Hay algún aspecto que le gustaría mencionar sobre el uso de UCIntlm que no se haya tratado en esta encuesta? Si es así, por favor, diga de que se trata:

Anexo 2. Diagrama de Clases

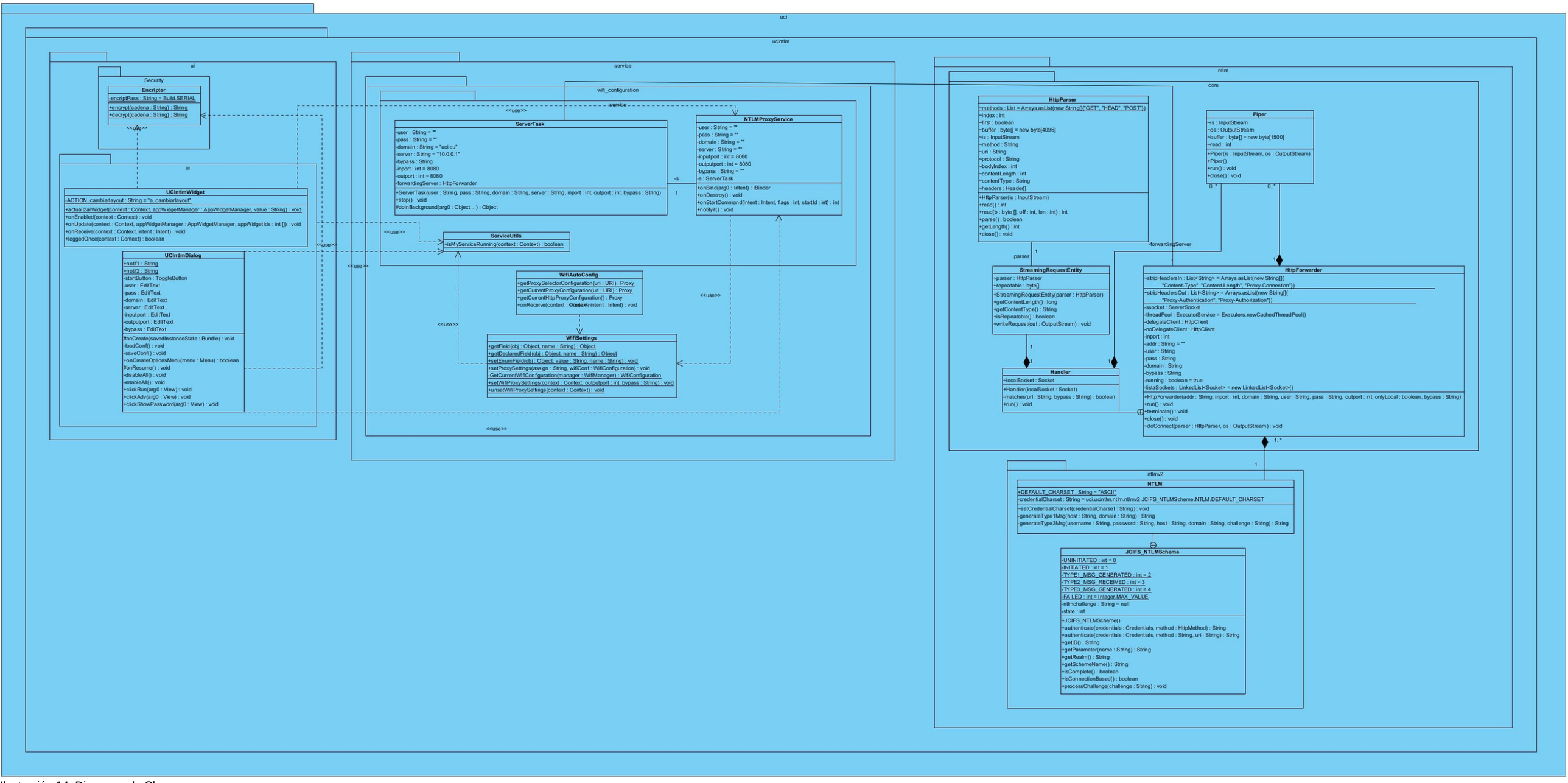

Ilustración 14: Diagrama de Clases