Universidad de las Ciencias Informáticas

Facultad 2

Herramienta software para la captura de tráfico IP en dispositivos móviles que utilicen Android como sistema operativo.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Dainé Sánchez Fuentes

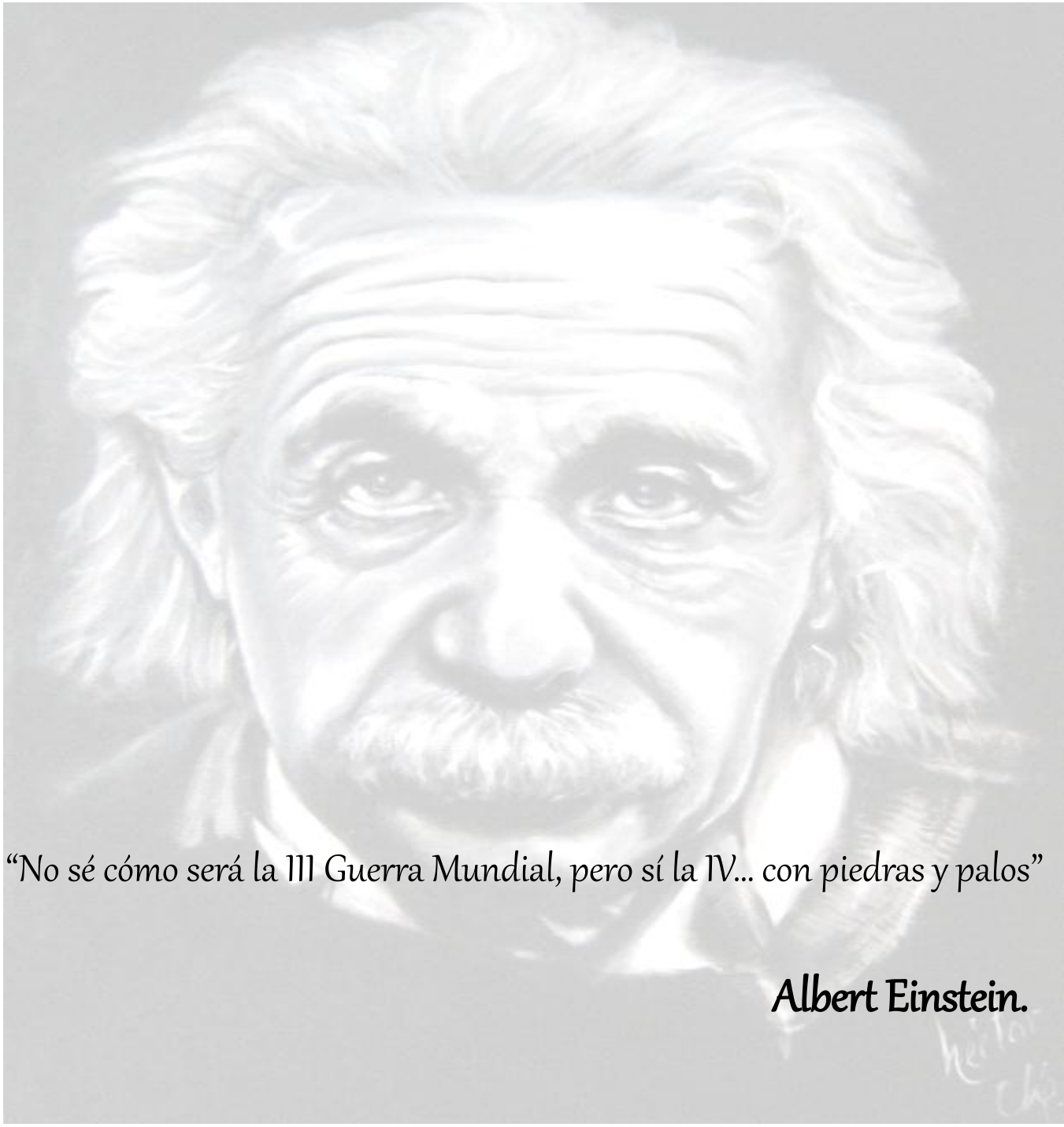
Alan Beltrán Graverán

Tutor:

MSc. Dayron Agüero Jiménez

Universidad de las Ciencias Informáticas, La Habana, 2013.

Pensamiento



"No sé cómo será la III Guerra Mundial, pero sí la IV... con piedras y palos"

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Dainé Sánchez Fuentes
Firma Autor

Alan Beltrán Graverán
Firma Autor

_____

Dayron Agüero Jiménez
Firma Tutor

DATOS DE CONTACTO

daguero@uci.cu Msc. Dayron Agüero Jiménez, graduado de Ingeniera en Ciencias Informáticas. Pertenece al centro de TLM, Dpto. Práctica Profesional.

*A mi maravillosa **madre**, por ser mi luz y creer siempre en mí.*

*A mi **padre** por sus sabios consejos y todos sus regaños.*

*A mi querida **hermanita** por apoyarme siempre.*

*A mi familia por su apoyo incondicional en todo,
especialmente a mis tías **Gredy** y **Elena**.*

*A **Gabriel** por ser cómo mi segundo hermano y
haber soportado todas mis malacrianzas.*

*A **Rolando** por su ayuda y apoyo.*

*A todos mis amigos especialmente,
a **Eduardo** y a **Lien**.*

*A mi **compañero** de tesis.*

*A mi **tutor**.*

*A la **vida**.*

*Y a **mí**.*

Dainé Sánchez Fuentes.

*Quiero agradecer a la persona más especial de mi vida, **a mi mamá**, por toda la dedicación y cuidado formándome como persona, por apoyarme incondicionalmente en todos los momentos difíciles, por tenernos a mi hermano y a mí siempre presente por encima de sus necesidades, por todo el amor que me has brindado desde que abrí los ojos por primera vez, gracias por existir te quiero mucho.*

***A mi hermano**, recordarle que lo importante es salir adelante y que cuenta con mi ayuda para todo lo que necesite, espero ser ese hermano en el que puedas confiar y agradecerte por formar parte especial de mi vida.*

***A mi papá** por apoyarme en un día como hoy y ayudar que las personas a las que más amo estén hoy junto a mí.*

***A mi abuela Rosa y mi tía Carmita** por ser mis segundas madres, por sus sabios consejos y todo el cariño que me han brindado a lo largo de mi vida.*

***A mi primos Lázaro y Marta Rosa** por ser como hermanos para mí y considerarme de tal forma para ellos.*

***A mi abuelo Servilio y a mi padrino Roberto** por ser mis segundos padres y apoyarme en todo el trascurso de mi vida.*

***A mi primo Leonardito y a su mujer** por su preocupación en todo el entorno de mi carrera y ayudarme en el momento más importante cuando pensé que no iba a poder seguir. Gracias por la tecnología.*

A mi novia Lisi *que desde que nos conocimos ha estado ahí siempre para ayudarme en cualquier problema que tuviese, sin pedir nada a cambio, solo cariño, por darme todo su amor, por demostrarme que si se puede que solo hay que intentarlo, por ser en estos momentos una gran parte de mi vida de cual no me quiero separar nunca, gracias por existir por ser mi otra mitad. Te Amo.*

Agradecer al piquete de amigos del edificio 15, ***Alejandro, Michel, Ángel, Linares, Rafael, Leo, Luis, Souley, Denis, Luis Olfride, Dalien, Javier, Junier, Darien, Sergio, José Luis, Jonathan, Arián, Reinier*** *que de una manera u otra han estado al pendiente de mis problemas en mi carrera, por el apoyo brindado en los momentos difíciles y por todo lo que hemos hecho juntos en todos estos años de amistad.*

A mis compañeros de aula que están conmigo desde primer año porque de una forma u otra Uds. ayudaron a que llegara hasta aquí, ***a Suanny, Kati, Lien, Irma, Danisei*** *y los varones ya mencionados anteriormente.*

A mi compañera de tesis Dainé *por ayudarme todo este tiempo que hemos estado juntos trabajando para lograr un mismo propósito graduarnos.*

A todos los presentes por darme su apoyo, a los que de una forma u otra han ayudado a mi formación como profesional, ***a mi tutor*** *por ayudarnos en lo posible en el desarrollo de la tesis. Gracias a todos.*

Alan Beltrán Graverán.

A mi madre, a mi padre, a mi hermana y mis sobrinitos.

Dainé Sánchez Fuentes.

***A mi mamá** por creer en mí todo este tiempo, por esperar pacientemente por que alcanzara mis metas, por darme todo el cariño del mundo, por ser mi gran tesoro, te quiero mamá.*

Alan Beltrán Graverán.

Resumen

En los últimos 10 años la industria de las comunicaciones móviles ha crecido de forma espectacular. Las telecomunicaciones han evolucionado hacia redes de comunicaciones, cuyo objetivo es la disponibilidad de servicios de excelencia gracias a internet móvil. En este sentido los teléfonos inteligentes se están convirtiendo en una plataforma potente ya que están diseñados para estar permanentemente consumiendo estos servicios desde internet. En este nuevo escenario es necesario la verificación de las prestaciones de los servicios y el monitoreo del flujo de datos que trasmiten y reciben las múltiples aplicaciones de los dispositivos móviles. En el presente trabajo se desarrolla una herramienta que permite monitorizar este tráfico generado.

Para llegar a esta solución se analizan sistemas similares identificando las características útiles que se puedan incorporar al desarrollo de la solución. Se seleccionan las herramientas y tecnologías idóneas y se guía el proceso de desarrollo conforme a lo establecido por la metodología XP. Como resultado de la investigación se obtiene una herramienta que permite capturar tráfico IP para los teléfonos inteligentes con sistema operativo Android en su versión 2.2.

Palabras claves: Android, captura, redes, telefonía móvil, tráfico IP.

Índice

Índice de Tablas

Introducción

La tecnología ha avanzado rápidamente a lo largo de los años modernizando la comunicación entre los seres humanos. Dentro de estos grandes logros se encuentra Internet, la cual hace posible el acceso global y económico a un mundo de información, entretenimiento y conocimiento. El crecimiento de internet en lo que respecta a infraestructuras así como al número de usuarios conectados ha sido vertiginoso en los últimos tiempos. Este crecimiento ha dado lugar al surgimiento de un conjunto de nuevos servicios y aplicaciones tales como las comunicaciones multimedia o el comercio electrónico.

En este nuevo escenario las redes inalámbricas adquieren un éxito sin precedentes debido a una combinación de factores: una tecnología eficaz con el uso del espectro electromagnético[1], orientada al despliegue de redes locales de pequeño tamaño, un entorno regulatorio que permite su libre uso, una lógica fácilmente integrable y de muy bajo coste y una interoperabilidad de equipos generalmente exitosa (1). Las redes inalámbricas han adquirido mucho auge y un gran avance en estos últimos años. Una de las áreas en las que ha tenido gran desarrollo, ha sido la telefonía celular con el progreso de la telefonía móvil digital.

Inicialmente la telefonía celular se concibió para las comunicaciones verbales, pero es en la actualidad cuando el desarrollo de las telecomunicaciones ha originado un crecimiento constante y acelerado de suscriptores, que cada vez se vuelven más exigentes en cuanto a tecnología de alta capacidad de transmisión no sólo de voz sino también de datos con posibilidades de incluir nuevos servicios de valor agregado; esto ha traído como consecuencia el desarrollo de ideas y sistemas innovadores más sofisticados con respecto al concepto de brindar un servicio de excelencia a los abonados (2).

Los teléfonos inteligentes, especialmente los que utilizan el sistema operativo Android, están diseñados para sacar el máximo partido a Internet a través de una gran experiencia móvil. Estos dispositivos están colmados de las mejores aplicaciones de Google adaptadas para internet móvil como Google Maps y Navigation, Google Talky Gmail. Además poseen aplicaciones ya pre-cargadas de los servicios más conocidos de Internet como Facebook, YouTube, Twitter y MySpace. Debido a que los dispositivos

[1] Rango de ondas que se puede utilizar para transmitir información.

inteligentes están diseñados para estar conectados permanentemente a Internet, existen usuarios administradores de redes, analistas de redes o especialistas de redes que necesitan saber porque la conexión se realiza tan lenta o porque el dispositivo no se conecta a la red. También necesitan experimentar o comprobar cómo funcionan ciertos protocolos de red y examinar con detalle los paquetes que están trasmitiendo y recibiendo constantemente algunas aplicaciones y servicios en la red móvil de datos. Además usuarios que estén desarrollando protocolos, o cualquier programa que transmita y reciba datos en una red, precisan comprobar lo que realmente hace el programa.

A partir de la **problemática** descrita anteriormente se define el siguiente **problema a resolver**: ¿Cómo monitorizar el tráfico generado por las aplicaciones presentes en los teléfonos móviles que utilicen Android 2.2 como sistema operativo?

Para el desarrollo de la presente investigación se ha centrado el **objeto de estudio** en los procesos asociados a la captura de tráfico IP.

Para dar solución al problema planteado se traza como **objetivo general**: desarrollar una herramienta software que permita la captura de tráfico IP[2] en teléfonos móviles que utilicen Android 2.2 como sistema operativo.

Se enfoca el **campo de acción** en los procesos asociados a la captura de tráfico IP para teléfonos móviles con sistema operativo Android 2.2.

Para darle sustento al objetivo general se definen los siguientes **objetivos específicos**:

- ✓ Implementar una herramienta que capture tráfico IP en teléfonos móviles que utilicen Android 2.2 como sistema operativo.
- ✓ Devolver la captura del tráfico IP en un archivo que contenga un formato legible por el analizador de protocolos WireShark[3].

[2] Protocolo Internet
[3] Es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica para educación.

- ✓ Probar la herramienta obtenida en un emulador, tratando de garantizar un ambiente lo más real posible.

Para darle cumplimiento al objetivo general se proponen las siguientes **tareas de la investigación**:

- ✓ Análisis del marco de trabajo para desarrollar en Android.
- ✓ Realización del marco teórico de la investigación para garantizar el basamento y fundamento científico de la misma.
- ✓ Análisis y Diseño de la herramienta de captura de tráfico IP para garantizar un punto de partida de la implementación.
- ✓ Análisis de los formatos de entrada del analizador de protocolos WireShark para determinar el formato a utilizar en la captura de las tramas IP.
- ✓ Definición de la estructura del formato a utilizar en la captura de las tramas IP para que pueda establecerse la interacción con el analizador de protocolos WireShark.
- ✓ Realización de la herramienta de captura de tráfico IP para garantizar el correcto uso de los recursos y protocolos en los teléfonos móviles.
- ✓ Validación de la herramienta en un emulador para garantizar su correcto funcionamiento.

Métodos de la investigación:

Analítico-Sintético: Centrándose en el análisis de las teorías, documentos, y materiales, permite la extracción de los elementos más importantes para realizar el análisis y estudio relacionado a la captura de tráfico IP.

Histórico-Lógico: Se logrará una mayor comprensión del estado actual de las herramientas de captura de tráfico IP a partir del análisis de su evolución y las etapas principales por las que han transitado, tomar lo positivo de los sistemas que presenten características comunes para una buena fundamentación y posterior implementación.

El presente documento está compuesto por 4 capítulos:

En el **Capítulo 1** “Fundamentación Teórica” se exponen los fundamentos generales que sirven de soporte teórico en la solución del problema. Se analizan las herramientas y lenguajes de programación idóneas para la implementación de la herramienta. Además se plantea la metodología a emplear en el desarrollo de la misma.

En el **Capítulo 2** “Características del Sistema. Exploración y Planificación” se describen las características del sistema a desarrollar. Se da inicio a las fases de Exploración y Planificación. Se define lo que debe hacer el sistema a partir de las funcionalidades requeridas y las historias de usuario impuestas por el cliente.

En el **Capítulo 3** “Diseño del sistema” se define la arquitectura del sistema y con ello los patrones arquitectónicos empleados. Además para una adecuada organización de las clases principales, se emplean las tarjetas Clase – Responsabilidad – Colaborador (CRC), modelo que describe las funcionalidades del sistema.

En el **Capítulo 4** “Implementación y Pruebas” se exponen las principales características de la implementación. Se procede a desarrollar las tareas de la ingeniería que responden a las Historias de Usuario (HU) abordadas en cada iteración, luego mediante las pruebas se verifica que el producto resultante cumpla con los requerimientos definidos.

Capítulo 1: Fundamentación Teórica

Introducción al capítulo

En este capítulo se abordan algunos aspectos del desarrollo de las redes de comunicaciones de datos, así como los principios básicos necesarios para soportarlas. También se repasan los conceptos asociados la captura de tráfico IP. Finalmente, se exponen las tecnologías y la metodología que se utiliza en el desarrollo de la aplicación.

1.1 Redes

Una red de computadora o red de comunicación de datos es un conjunto de dispositivos autónomos interconectados y se dice que dos dispositivos están interconectados si pueden intercambiar información (3).

Las redes de datos, tienen como objetivos:

- ✓ Compartir recursos, equipos, información y programas que se encuentran localmente o dispersos geográficamente.
- ✓ Brindar confiabilidad a la información, disponiendo de alternativas de almacenamiento.
- ✓ Obtener una buena relación coste / beneficio.
- ✓ Transmitir información entre usuarios distantes de la manera más rápida y eficiente posible.

Básicamente, las redes sirven para comunicarnos y la comunicación posibilita el intercambio fluido de información, proceso que permite su almacenamiento, procesamiento, uso y distribución. Al mismo tiempo, todo esto es imprescindible para evolucionar en los diversos aspectos de nuestra civilización (4).

1.2 Protocolos

Para establecer la comunicación entre las computadoras se requiere de un lenguaje común: el protocolo. Los protocolos son estándares de software que se instalan en las computadoras de una red para definir el lenguaje, las reglas, los procedimientos y las metodologías utilizadas, para que las máquinas de la red puedan entenderse entre ellas. El uso de protocolos permite a las computadoras comunicarse,

intercambiar información y atender errores que puedan producirse durante el intercambio. En cierto sentido los protocolos son para las comunicaciones lo que los algoritmos para la computación. Un algoritmo permite especificar o entender un cómputo aunque no conozca los detalles de un juego de instrucciones de CPU. De manera similar, un protocolo de comunicaciones permite especificar o entender la comunicación de datos sin depender de un conocimiento detallado de una marca en particular de hardware de red (5)**.**

Existen muchos estándares para protocolos de comunicación pero TCP/IP es el lenguaje universal que usan las computadoras y redes para comunicarse entre sí.

TCP/IP es un grupo de protocolos diseñados para la comunicación entre computadoras suministrando, a su vez, servicios de red como: registro de entrada remoto, transferencia remota de archivos, correo electrónico, etc. Un protocolo de comunicación debe manejar los errores en la transmisión de datos, administrar el enrutamiento y entregar los datos así como controlar la transmisión real de red mediante el uso de señales de estado predeterminadas y TCP/IP se ocupa de todo esto (6)**.**

Se emplea en internet y constituye en la actualidad una forma sumamente importante de tecnologías para redes. Se basa en el concepto de cliente/servidor: cualquier dispositivo que inicie una comunicación se llama cliente y el dispositivo que responde, servidor (6)**.**

1.3 Arquitectura del Modelo TCP/IP

El protocolo TCP/IP fue creado antes que el modelo de capas OSI, así que los niveles del protocolo TCP/IP no coinciden exactamente con los siete que establece el OSI. Los datos que son enviados a la red recorren la pila del protocolo TCP/IP desde la capa más alta de aplicación hasta la más baja de enlace. Cuando son recibidos, recorren la pila de protocolo en el sentido contrario. Durante estos recorridos, cada capa añade o sustrae cierta información de control a los datos para garantizar su correcta transmisión (Figura 1) (7).

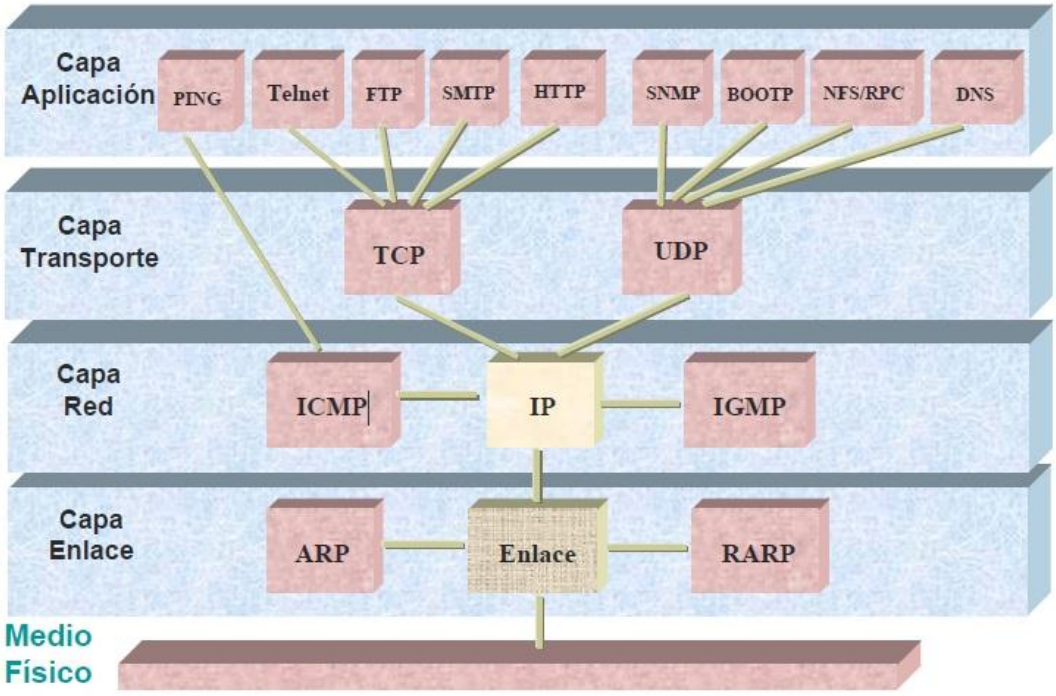


Figura 1: Modelo TCP/IP.

En teoría cada capa maneja una estructura de datos propia, independiente de las demás, aunque en la práctica estas estructuras de datos se diseñan para que sean compatibles con las de las capas adyacentes (7).

1.3.1 Capa de Enlace

Dentro de la jerarquía del protocolo TCP/IP la capa de enlace se encuentra en el nivel más bajo. La capa de enlace es el mecanismo que utiliza un ordenador para conectarse a la red. Esta capa maneja todos los aspectos que un paquete IP requiere para efectuar un enlace físico real con los medios de la red. La capa de enlace define los procedimientos para realizar la interfaz con el hardware de la red y para tener acceso al medio de transmisión. Sus funciones son: la asignación de direcciones IP a las direcciones físicas, el encapsulamiento de los paquetes IP en tramas (7).

Dentro de esta capa opera el Protocolo de Resolución de Dirección (ARP, por sus siglas en ingles), que se encarga precisamente de asociar direcciones IP con direcciones físicas Ethernet. El protocolo de

Resolución de Dirección permite a una máquina conocer sobre la actividad del protocolo de nivel superior en un cable Ethernet. Puede determinar que campos de tipo de protocolo Ethernet están en uso y las direcciones de protocolo en cada tipo de protocolo. (8)

1.3.2 Capa de Red

La capa de red se encuentra encima de la capa de enlace, esta se encarga de encaminar los datos a través de las distintas redes hasta llegar a su destino (7). En este nivel se encuentra el protocolo IP, encargado del intercambio de paquetes entre ordenadores por la red. En esta capa también se encuentra el Protocolo de Mensajes de Control Internet (ICMP, por su definición en inglés) que es un protocolo de supervisión empleado para el envío de mensajes en forma de datagramas de los errores o incidencias que ocurren en la transferencia del datagrama IP. Los mensajes ICMP son enviados en varias situaciones: por ejemplo, cuando un datagrama no puede alcanzar su destino, cuando una pasarela no dispone de capacidad de almacenamiento temporal para reenviar el datagrama y cuando la pasarela puede dirigir al "ordenador" para enviar el tráfico por una ruta más corta (9). También interactúa en esta capa junto al protocolo IP e ICMP el Protocolo de Administración de Grupos de Internet (IGMP, por sus siglas en inglés). Este protocolo es utilizado por los sistemas IPv4 para informar a sus miembros en el grupo de multidifusión[4] IP los routers de multidifusión vecinos.

1.3.3 Capa de Transporte

La capa de transporte proporciona servicios de transporte desde el ordenador origen hacia el ordenador destino. En esta capa se forma una conexión lógica entre los puntos finales de la red, el ordenador transmisor y el ordenador receptor. Los protocolos de transporte segmentan y re ensamblan los datos mandados por las capas superiores en el mismo flujo de datos, o conexión lógica entre los extremos. La corriente de datos de la capa de transporte brinda transporte de extremo a extremo. En esta capa operan el Protocolo de Control de Transmisión (TCP, pos sus siglas en inglés) y el Protocolo de Datagramas de Usuario (UDP, por sus iniciales en inglés) (7).

[4] Es el envío de la información en una red a múltiples destinos simultáneamente.

El protocolo TCP está pensado para ser utilizado como un protocolo 'ordenador' a 'ordenador' muy fiable entre miembros de redes de comunicación de computadoras por intercambio de paquetes y en un sistema interconectado de tales redes. Encaja en una arquitectura de protocolos en capas justo por encima del protocolo de internet, protocolo básico que proporciona un medio para TCP de enviar y recibir segmentos de longitud variable de información envuelta en "sobres" de datagramas de internet. El datagrama de internet proporciona un medio de direccionar TCP de origen y de destino situados en redes diferentes. El Protocolo de Internet también trata con la fragmentación y el reensamble de segmentos de TCP que sean necesarios para conseguir el transporte y la entrega sobre múltiples redes y las puertas de enlace que las interconectan. El Protocolo de Internet también lleva información sobre la prioridad, clasificación de seguridad y compartimentación de los segmentos de TCP, de tal forma que esta información pueda ser comunicada de extremo a extremo entre múltiples redes (10).

El protocolo UDP se define con la intención de hacer disponible un tipo de datagramas para la comunicación por intercambio de paquetes entre ordenadores en el entorno de un conjunto interconectado de redes de computadoras. Este protocolo asume que el Protocolo de Internet se utiliza como protocolo subyacente. Este protocolo aporta un procedimiento para que los programas de aplicación puedan enviar mensajes a otros programas con un mínimo de mecanismo de protocolo. El protocolo se orienta a transacciones, y tanto la entrega como la protección ante duplicados no se garantizan (11).

1.3.4 Capa de Aplicaciones

La capa de aplicación del modelo TCP/IP maneja protocolos de alto nivel, aspectos de representación, codificación y control de diálogo. El modelo combina todos los aspectos relacionados con las aplicaciones en una sola capa y asegura que estos datos estén correctamente empaquetados antes de que pasen a la capa siguiente. Funciona de la siguiente manera: los usuarios llaman a una aplicación, la aplicación interactúa con uno de los protocolos de nivel de transporte para enviar o recibir datos. Cada programa de aplicación selecciona el tipo de transporte necesario, el cual puede ser una secuencia de mensajes individuales o un flujo continuo de octetos (7).

Operan en esta capa:

El Protocolo de Terminal de Red (Telnet*, Network Terminal Protocol*) que permite establecer conexiones con terminales remotos, de tal manera que se puedan ejecutar en ellos comandos de configuración y control. El propósito del protocolo Telnet es proporcionar un servicio de comunicaciones orientado a bytes de 8 bit general y bidireccional. El principal objetivo es permitir un método estándar de comunicar entre sí terminales y procesos orientados a terminal (12).

El Protocolo de Transferencia de Ficheros (FTP, por sus siglas en inglés) promociona el uso compartido de ficheros (programas y/o datos), anima al uso indirecto o implícito (a través de programas) de servidores remotos, hace transparente al usuario las variaciones entre la forma de almacenar ficheros en diferentes ordenadores, transfiere datos fiable y eficientemente. Este protocolo aunque puede ser utilizado directamente por un usuario en un terminal, está diseñado principalmente para ser usado por programas (13).

El protocolo SMTP (Simple Mail Transfer Protocol, Protocolo Simple de Transferencia de Correo) tiene como objetivo transferir correo fiable y eficiente. Posibilita el funcionamiento del correo electrónico en las redes de ordenadores, este protocolo recurre al Protocolo de Oficina Postal (POP, pos sus siglas en inglés) para almacenar mensajes en los servidores de correo electrónico, existen dos versiones: POP2**,** que necesita la intervención de SMTP para enviar mensajes y POP3 que funciona de forma independiente (14).

El Protocolo Transferencia de Hipertexto (HTTP, *Hipertext Transfer Protocol*) es un protocolo de nivel de aplicación de los sistemas de información hipermedia distribuidos y colaborativos. HTTP es un protocolo de solicitud / respuesta. Un cliente envía una solicitud al servidor en la forma de un método de petición, URI (Localizador de Recursos Uniforme, *Uniform Resource Locator*) y la versión del protocolo, seguido por un mensaje que contiene una demanda de modificadores, la información del cliente, y el posible contenido del cuerpo través de una conexión con un servidor. El servidor responde con una línea de estado, incluyendo la versión de protocolo del mensaje y un código de éxito o de error, seguido por un mensaje que contiene información del servidor, entidad de información de metadatos, y el posible contenido entidad-cuerpo (15). HTTP permite la transmisión de gran variedad de archivos de texto,

gráficos, sonidos e imágenes, regula el proceso mediante el cual navegadores como Netscape, Mozilla o Internet Explorer solicitan información a los servidores web.

1.4 Protocolo Internet

El protocolo internet (conocido por sus siglas en inglés IP, Internet Protocol) es parte de la suite de protocolos TCP/IP de internet. Está diseñado para su uso en sistemas interconectados de redes de comunicación de ordenadores por intercambio de paquetes. El Protocolo Internet está específicamente limitado a proporcionar las funciones necesarias para enviar un paquete de bits (un datagrama internet) desde un origen a un destino a través de un sistema de redes interconectadas. No existen mecanismos para aumentar la fiabilidad de datos entre los extremos, control de flujo, u otros servicios que se encuentran normalmente en otros protocolos ordenador-a-ordenador (16)**.**

Implementa dos funciones básicas: direccionamiento y fragmentación. La primera usada para transmitir los datagramas hacia sus destinos que consiste en mover datagramas internet a través de redes interconectadas. Los datagramas son encaminados desde un módulo internet a otro a través de redes individuales basándose en la interpretación de una dirección internet. La segunda se encarga de re-ensamblar y fragmentar los datagramas cuando necesiten atravesar una red cuyo tamaño máximo de paquete es menor que el tamaño del datagrama.

Datagrama internet (16):

El datagrama, se divide en dos campos: cabecera y datos. La cabecera contiene información que el protocolo necesita para ofrecer su servicio, y el campo de datos contiene el mensaje en sí que tiene que ser entregado en el ordenador destino. La Figura 2 muestra el formato de la cabecera del datagrama IP.

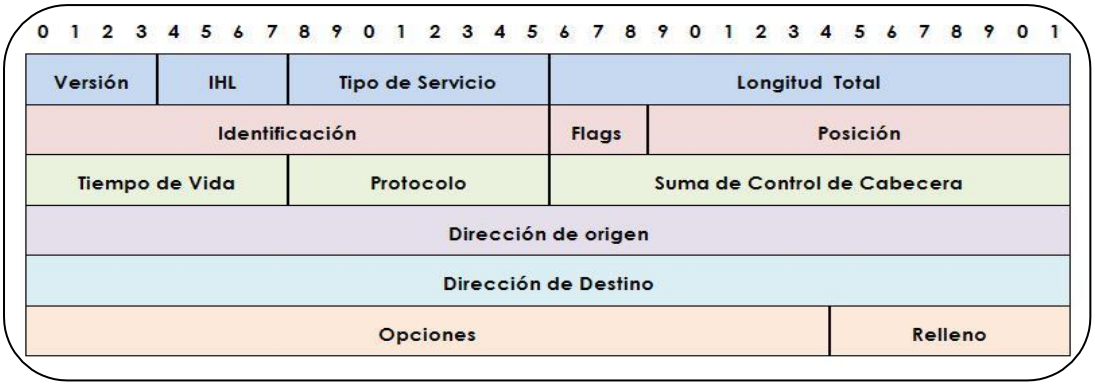


Figura 2: Datagrama IP.

Descripción de los campos (16):

- ✓ **Versión 4 bits:** el campo Versión describe el formato de la cabecera internet. Especifica la versión del protocolo IP a la que pertenece el datagrama.

- ✓ **IHL 4 bits:** Longitud de la Cabecera Internet (Internet Header Length), es la longitud de la cabecera en palabras de 32 bits, y por tanto apunta al comienzo de los datos.

- ✓ **Tipo de Servicio 8 bits:** indica de los parámetros abstractos de la calidad de servicio deseada. Estos parámetros se usarán para guiar la selección de los parámetros de servicio reales al transmitir un datagrama a través de una red en particular. Algunas redes ofrecen prioridad de servicio, la cual trata de algún modo el tráfico de alta prioridad como más importante que el resto del tráfico (generalmente aceptando sólo tráfico por encima de cierta prioridad en momentos de sobrecarga). La elección más común es un compromiso a tres niveles entre baja demora, alta fiabilidad, y alto rendimiento. Figura 3.

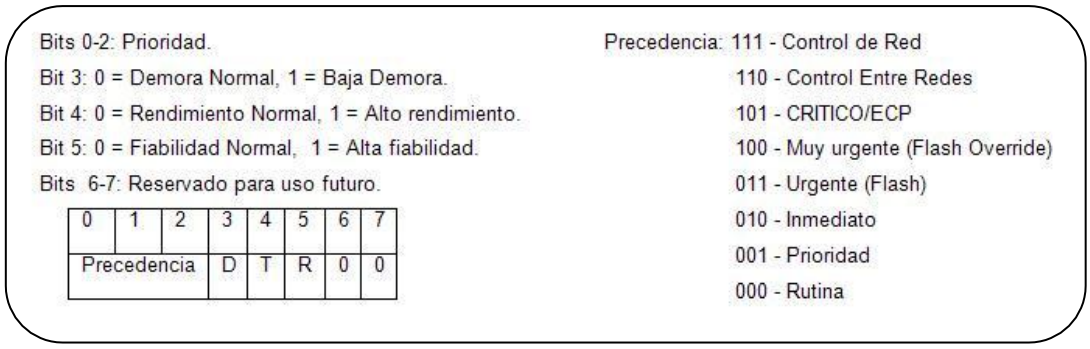


Figura 3: Valores del campo tipo de servicio.

- ✓ **Longitud Total 16 bits:** es la longitud del datagrama, medida en octetos, incluyendo la cabecera y los datos. Todos los ordenadores deben estar preparados para aceptar datagramas de hasta 576 octetos (tanto si llegan completos como en fragmentos). Este tamaño permite que un bloque de datos de 512 octetos más 64 octetos de cabecera quepa en un datagrama.

- ✓ **Identificación16 bits:** es un valor de identificación asignado por el remitente como ayuda en el ensamblaje de fragmentos de un datagrama.

- ✓ **Flags (indicadores) 3 bits:** son diversos indicadores de control. Bit 0: reservado, debe ser cero. Bit 1: no fragmentar (Don't Fragment, DF) si es 0 = puede fragmentarse, 1 = No Fragmentar. Bit 2: más fragmentos (More Fragments, MF) si es 0 = Último Fragmento, 1 = Más Fragmentos.

- ✓ **Posición del Fragmento 13 bits:** este campo indica a que parte del datagrama pertenece este fragmento. La posición del fragmento se mide en unidades de 8 octetos (64bits). El primer fragmento tiene posición 0.

- ✓ **Tiempo de Vida 8 bits:** este campo indica el tiempo máximo que el datagrama tiene permitido permanecer en el sistema internet. Si este campo contiene el valor cero, entonces el datagrama debe ser destruido. Este campo es modificado durante el procesamiento de la cabecera internet. El tiempo es medido en segundos, todo módulo que procese un datagrama debe decrementar el TTL (Time To Live: Tiempo de Vida) al menos en uno.

- ✓ **Protocolo 8 bits:** este campo indica el protocolo del siguiente nivel usado en la parte de datos del datagrama internet. Los valores de varios protocolos son especificados en "Números Asignados".

- ✓ **Suma de Control de Cabecera 16 bits:** dado que algunos campos dela cabecera cambian (p. ej. el tiempo de vida), esta suma es re-calculada y verificada en cada punto donde la cabecera internet es procesada. El algoritmo de la suma de control es el complemento a uno de 16 bits de la suma de los complementos a uno de todas las palabras de 16 bits de la cabecera. A la hora de calcular la suma de control, el valor inicial de este campo es cero.

- ✓ **Dirección de Origen 32 bits:** dirección IP origen.

- ✓ **Dirección de Destino 32 bits:** dirección IP destino.

- ✓ **Opciones (variable):** las opciones pueden o no aparecer en los datagramas. Deben ser implementadas por todos los módulos IP (ordenadores y pasarelas). Lo que es opcional es su transmisión en cualquier datagrama en particular, no su implementación. El campo opción es de longitud variable. Pueden existir cero o más opciones.

- ✓ **Valor de Relleno (variable):** se usa para asegurar que la cabecera internet ocupa un múltiplo de 32 bits. El valor de relleno es cero.

1.5 Tráfico de red

El tráfico de red es la cantidad de información promedio que se transfiere a través del canal de comunicación. En concreto el tráfico de red son los datos que se transmiten para luego ser procesados.

En un principio la transmisión de datos era analógica pero al aumentar los servicios fue necesario entonces adaptar la transmisión a señales digitales. Los datos pueden ser transmitidos en diversas velocidades.

1.6 Captura y Análisis de tráfico IP

Capturar tráfico de red es la acción de coleccionar los datos que se transfieren a través del canal de comunicación. Se basan en la captura o registro de la información contenida en la trama o datagrama IP que se transfiere por un segmento red.

Una vez capturados, los paquetes entregan información sobre el sentido del flujo (origen-destino), cantidad de información transferida, protocolos empleados. El análisis posterior de la información que se transfiere a través de las redes y/o sus enlaces y la búsqueda posterior de patrones o características que muestren alguna tendencia o comportamiento es lo que se conoce como análisis de tráfico de red.

El objetivo de capturar y analizar el tráfico de red es realizar un seguimiento y análisis de protocolos. Además se puede determinar el tipo de información que circula por la red y el impacto que pudiera llegar a tener sobre la misma.

1.7 Sistemas Operativos

Los sistemas operativos son una parte esencial de cualquier sistema informático. Un sistema operativo es un programa que administra el hardware de un sistema informático. Proporciona los mecanismos apropiados para asegurar el correcto funcionamiento del sistema informático e impedir que los programas de usuario interfieran con el apropiado funcionamiento del sistema. Algunos sistemas operativos se diseñan para ser prácticos, otros para ser eficientes y otros para ambas cosas (17).

Existe una enorme y variada gama de sistemas informáticos para los que se diseñan sistemas operativos (18). Tal es el caso de los teléfonos móviles que tratan de llevar más lejos el concepto de telefonía añadiendo a los terminales móviles funciones propias de los ordenadores. Los dispositivos móviles de la actualidad proporcionan una serie de servicios basados en el tráfico de datos a través de la red, por lo que necesitan un sistema operativo propio a este tipo de escenario.

Existen varios sistemas operativos para dispositivos móviles de los cuales se puede citar Windows Phone, desarrollado por Microsoft y lanzado al mundo el 15 de febrero del 2010. Symbian OS que es un producto

de varias empresas[5] y fue lanzado al mercado en 1997, se implementa principalmente en dispositivos Nokia aunque ha sido implementado en Samsung, LG, Lenovo y otros dispositivos de otros fabricantes. Además se encuentra el iOS un producto de la empresa Apple Inc., lanzado el 9 de enero del 2007 y solo es usado en dispositivos de este fabricante. BlackBerry OS lanzado en 1999 por la empresa Research In Motion para sus dispositivos.

1.7.1 Android

Es una plataforma desarrollada por Google y el consorcio Handset Alliance[6]. Presenta una serie de características que lo hacen diferente (19):

- ✓ Plataforma realmente abierta: Es una plataforma de desarrollo libre basada en Linux y de código abierto.
- ✓ Portabilidad asegurada: las aplicaciones finales son desarrolladas en Java lo que asegura que podrán ser ejecutadas en gran variedad de dispositivos.
- ✓ Arquitectura basada en componentes basada en internet.
- ✓ Filosofía de dispositivo siempre conectado a internet.
- ✓ Alto nivel de seguridad: Los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que incorpora la máquina virtual.
- ✓ Optimización para baja potencia y poca memoria: Android utiliza la máquina virtual Dalvik, que es una implementación de la máquina virtual de Java para dispositivos móviles.

Android, al contrario que otros sistemas operativos para dispositivos móviles como iOS o Windows Phone, se despliega de forma abierta y se permite el acceso tanto al código fuente como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos. Se usa en disimiles dispositivos como son en teléfonos inteligentes, ordenadores portátiles, notebooks, tabletas, Google TV, relojes de pulsera, auriculares y otros.

Arquitectura de Android (19):

[5] Estas empresas son: Nokia, Sony Mobile Communications, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc.
[6] Está formado por Google, Intel, Texas Instrument, Motorola, T-Mobile, Samsung, Ericson, Toshiba, Vodafone, NTT DoCoMo,Sprint Nextel y otros.

La figura 4 muestra la arquitectura de Android. La misma está formada de cuatro capas, una delas características más importantes es que todas las capas están basadas en software libre.

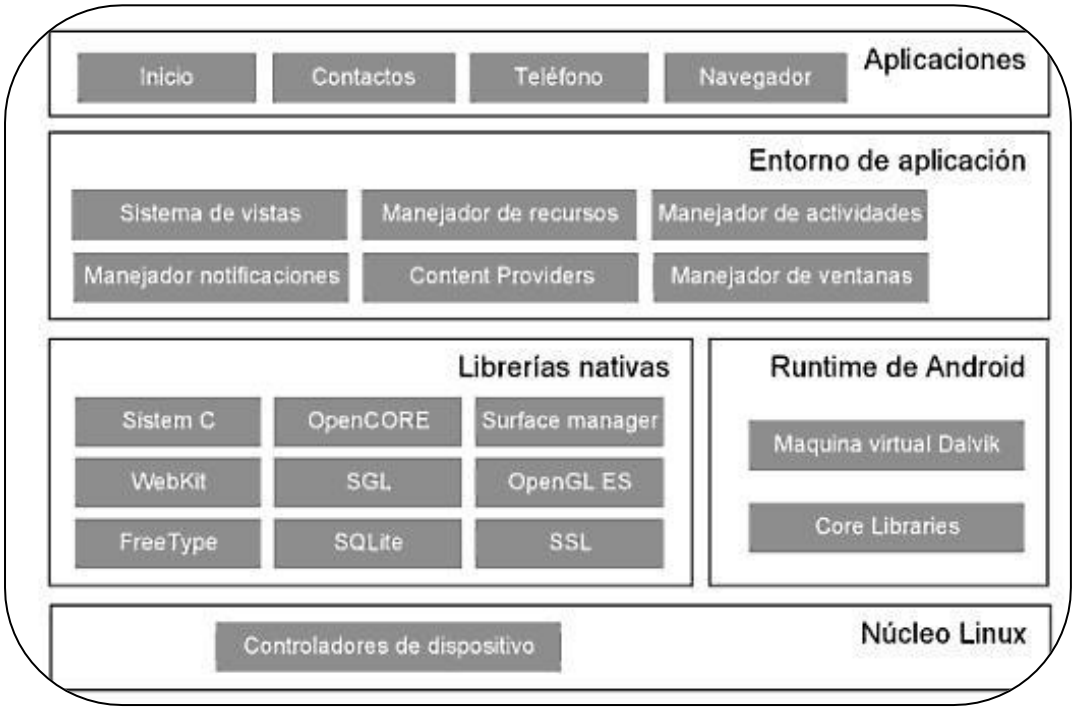


Figura 4: Arquitectura de Android.

Descripción de las capas (19):

1. Núcleo Linux:

El núcleo de Android está formado por el sistema operativo Linux versión 2.6. Esta capa proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos. Esta capa del modelo actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del hardware.

2. Entorno de Ejecución(Runtime)de Android:

Dado a las limitaciones de los dispositivos donde debe correr el sistema operativo android (poca memoria y procesador limitado) no fue posible utilizar la máquina virtual de Java. Google tomo la decisión de crear una nueva, la máquina virtual Dalvik, que responde mejor a estas limitaciones.

Algunas características de la máquina virtual Dalvik que facilitan la optimización de recursos son: que ejecuta ficheros Dalvik ejecutables (.dex)-formato optimizado para ahorrar memoria. Además está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al núcleo de Linux algunas funciones como el manejo de la memoria a bajo nivel.

3. Librerías nativas:

Incluye un conjunto de librerías C/C++ usadas en varios componentes de Android. Están compiladas en el código nativo del procesador. Algunas de estas librerías son:

- ✓ System C library: una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
- ✓ Media Framework: librería basada en PacketVideo's [7] Open CORE; soporta codecs [8] de reproducción y grabación de multitud de formatos de audio, video e imágenes.
- ✓ Web Kit: soporta el moderno navegador web utilizado en el navegador de Android y el vista webview.
- ✓ Librerías 3D: Implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador hardware 3D si está disponible, o el software altamente optimizado de proyección 3D.
- ✓ SQLite: potente y ligero motor de Bases de Datos relacionales disponible para todas aplicaciones.
- ✓ SSl: proporciona servicios de encriptación de capa de conexión segura.

Estas librerías normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma “más eficiente”.

[7] ES una empresa en San Diego, que produce software para móviles multimedia, incluyendo la visualización de vídeo en teléfonos móviles.

[8] Es la abreviatura de codificador-decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos o una señal.

4. Entorno de Aplicación:

Proporciona una plataforma de desarrollo libre con gran riqueza e innovaciones. Esta capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik.

5. Aplicaciones:

Este nivel está formado por el conjunto aplicaciones instaladas en una máquina Android, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen pre-instaladas en el dispositivo y aquellas que el usuario ha instalado. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema.

1.8 Estado del Arte

En este epígrafe se presentan algunas de las soluciones que existen en el mundo al problema en cuestión. Se expresa además porque estas soluciones no pueden ser utilizadas.

1.8.1 SymPA

Es un analizador de protocolos que se ejecuta en el terminal y que permite capturar el tráfico TCP/IP entrante sin alterar el comportamiento de otras aplicaciones ni el resto de la funcionalidad del terminal. Tiene en cuenta los siguientes objetivos (20):

- ✓ Captura de todos los paquetes IP entrantes, asegurando que no se deja de analizar tráfico por problemas de saturación.
- ✓ Mantiene un consumo de recursos bajo, teniendo en cuenta las limitaciones de CPU y memoria de los dispositivos.
- ✓ Incorpora funciones básicas de generación de tráfico, como el test[9] de conectividad mediante ping[10].

[9] Una prueba o examen.

- ✓ Proporciona un interfaz que permita procesar la información capturada y exportarla a otros entornos.

La herramienta SymPA se ha desarrollado para el sistema operativo Symbian OS y la plataforma Serie 60 de Nokia (20). Esta herramienta no puede ser utilizada como una posible solución ya que no es compatible con el sistema operativo Android.

1.8.2 TestelDroid

TestelDroid es una herramienta de software para monitorizar los dispositivos basados en Android que permite la caracterización de las comunicaciones de datos en redes celulares. La información recuperada se puede registrar y exportar para su posterior análisis con otras herramientas, como WireShark. Todos los datos recogidos pueden ser registrados usando archivos planos de texto (excepto para la captura de tráfico, almacenados en formato pcap). El registro se implementa como un servicio de Android, por lo que puede estar en ejecución en segundo plano mientras realiza otras acciones en el teléfono. Los parámetros que se registran (red, células vecinas, batería, GPS, tráfico) son configurables. La herramienta ha sido probado en Nexus One (HTC), Nexus S (Samsung), Galaxy S (Samsung) y dispositivos Galaxy S3 (Samsung) con versiones de Android que van desde 2,2 (Froyo) a 4.1.1 (JellyBean) (21).

Esta herramienta fue creada por un pequeño grupo de desarrollo que exige un pago por el uso de la misma. Debido a que dicha herramienta es privada no se puede utilizar como solución del problema, por problemas económicos existente en el país para pagar por el uso de dicha herramienta.

1.9 Herramientas y metodologías

A continuación se exponen las herramientas y metodologías que serán utilizadas en el desarrollo de la herramienta.

[10] Como programa, ping es una utilidad que diagnóstica en redes de computadoras el estado de la comunicación con el host local con uno o varios equipos remotos de una red TCP/IP por medio del envío de paquetes ICMP de solicitud y de respuesta. Mediante esta utilidad puede diagnosticarse el estado, velocidad y calidad de una red determinada.

1.9.1 Eclipse Índigo 3.7

Eclipse es un entorno de desarrollo integrado de código abierto y multiplataforma. Tiene herramientas para desarrollar aplicaciones de consola, Web y Web Services. Da soporte a todo tipo de proyectos abarcando el ciclo de vida completo en el desarrollo de aplicaciones. Dicho IDE ha alcanzado un alto grado de madurez, así como más robustez y rendimiento (22).

Es soportado por los principales sistemas operativos. Continuamente se están desarrollando nuevos plugins[11] y revisando los anteriores. Eclipse permite la instalación de plugins destinados a mejorar las funcionalidades del propio IDE y a extenderse en más tecnologías (22).

Se selecciona este IDE porque resulta un entorno de desarrollo recomendable para trabajar con Android debido a que ha sido utilizado por los desarrolladores de Google para crear Android.

1.9.2 Lenguaje de programación: Java

Java es un lenguaje de programación y la primera plataforma informática creada por la empresa Sun Microsystems en 1995. Java se ejecuta en más de 850 millones de ordenadores personales de todo el mundo y en miles de millones de dispositivos, como dispositivos móviles y aparatos de televisión (23).

La creación de este lenguaje y plataforma se inspiró en las funcionalidades de otros lenguajes tales como C++, Eiffel, SmallTalk, Objetive C, Cedar/Mesa, Ada, Perl. El resultado es una plataforma y un lenguaje idóneo para el desarrollo de aplicaciones sencillas, orientadas a objetos, distribuidas, interpretadas, robustas, seguras, independientes de las arquitecturas, portables, eficaces, multitareas y dinámicas (23).

Se selecciona este lenguaje de programación porque permite optimizar el tiempo y el ciclo de desarrollo (compilación y ejecución). Además, tiene gran compatibilidad con el sistema operativo Android porque Dalvik, su máquina virtual es una optimización de la máquina virtual de Java. Cientos de aplicaciones desarrolladas en Java corriendo satisfactoriamente en la plataforma Android, certifican lo idóneo que resulta este lenguaje de programación.

[11] Es un programa o aplicación que añade funcionalidad al programa principal donde está hospedado.

1.9.3 Kit de Desarrollo de Software (SDK 15.0.1)

El SDK es el kit de desarrollo necesario para desarrollar aplicaciones utilizando Java como lenguaje de programación para el sistema operativo Android. Proporciona las bibliotecas API (Interfaz de Programación de Aplicaciones) y herramientas de desarrollo necesarias para crear, probar y depurar aplicaciones. Posee un simulador de teléfono basado en QEMU[12], documentación, ejemplos de código y tutoriales (24).

El SDK funciona en sistemas operativos como Windows XP, Vista o 7, Mac OS X 10.4.8 o superior y en Ubuntu 6.06 o superior, pudiendo también integrarlo con el entorno de desarrollo Eclipse (24).

1.9.4 Plugin ADT 15.0.1

Herramienta de Desarrollo de Android (ADT, por sus siglas en inglés) es un plugin para el IDE Eclipse que está diseñado para darle un ambiente potente, integrado en la construcción de aplicaciones de Android. Amplía las capacidades de Eclipse que permiten configurar rápidamente nuevos proyectos para Android, crear una interfaz de usuario de aplicación, agregar los paquetes basados en la API Framework Android, depurar sus aplicaciones utilizando las herramientas del SDK de Android (25).

El desarrollo de Eclipse con ADT es muy recomendable y es la manera más rápida para empezar. Con la configuración del proyecto guiada que ofrece, así como la integración de herramientas, editores de XML personalizados, y el panel de resultados de depuración, ADT le da un impulso increíble al desarrollo de aplicaciones Android (25).

1.9.5 Binario TCPdump

Es una herramienta en línea de comandos que tiene como función analizar el tráfico que circula por la red. Permite al usuario capturar y mostrar en tiempo real los paquetes transmitidos y recibidos en la red a la cual el ordenador está conectado. Está escrito por Van Jacobson, Craig Leres, y Steven McCanne que trabajaban en ese momento en el Grupo de Investigación de Red del Laboratorio Lawrence Berkeley (26).

[12] QEMU es un emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped). Tiene capacidades de virtualización dentro de un sistema operativo.

Usos de TCPdump (26):

- ✓ Para rastrear problemas en la red o para monitorear actividades de la misma.
- ✓ Realizar aplicaciones que midan el uso que se hace de la red, permitiendo realizar mejores planificaciones y afrontar posibles actualizaciones con una idea más aproximada a la realidad de la red.
- ✓ Diagnosticar el estado de la red y guardar los registros en ficheros, muy útil para detectar fallos o problemas de saturación.

Funciona en la mayoría de los sistemas operativos UNIX: Linux, Solaris, BSD, Mac OS X, HP-UX y AIX entre otros. En esos sistemas, tcpdump hace uso de la biblioteca libpcap para capturar los paquetes que circulan por la red (26).

1.9.6 Librería RootTools

RootTools es una librería que proporciona un conjunto estandarizado de herramientas para el desarrollo de aplicaciones que requieran acceso de súper usuario. Proporciona al desarrollador facilidad de uso, mejorara los tiempos de desarrollo y promueve la reutilización de código. Los desarrolladores pueden racionalizar los procesos, mejorar la eficacia de las aplicaciones y proporcionar una mejor experiencia para los usuarios.

1.9.7 Metodología: eXtreme Programming (XP)

La programación extrema es una metodología ligera, iterativa incremental, creada para desarrollar software en equipos pequeños y medianos que trabajan en proyectos con requerimientos difusos o cambiantes (27).

Se realizan pruebas constantemente del sistema y se le ofrece al cliente versiones continuas del mismo, esto provoca que se consigan productos usables con mayor rapidez que a su vez satisfacen las necesidades del usuario con mayor exactitud. El proceso de integración es continuo, por lo que el esfuerzo final para la integración es nulo. Posibilita integrar todo el trabajo con mucha facilidad. Gracias a la filosofía del “pair programming” (programación en parejas), se consigue aplicar las buenas prácticas que ofrecen esta metodología (27).

Se utiliza esta metodología ágil por ser extendida y documentada (27), se adapta a las necesidades de cualquier equipo de desarrollo. Exige que se establezca una comunicación más fluida con el cliente y que éste tenga mayor participación en el desarrollo del software, logrando que se involucre más en el proceso.

Conclusiones del capítulo

Con el desarrollo del marco teórico se logró organizar y guiar el trabajo hacia los objetivos específicos, profundizando en el estudio de algunos conceptos fundamentales para el desarrollo de una aplicación que permita capturar tráfico IP en dispositivos móviles que utilizan sistema operativo Android 2.2. Además se realizó un estudio de las aplicaciones similares existentes a nivel internacional, lo que posibilitó la selección de las principales características de las tecnologías, herramientas, lenguajes y técnicas de desarrollo propuestas para la solución del problema actual.

Capítulo 2: Características del Sistema. Exploración y Planificación

Introducción del capítulo

El presente capítulo se centra en las características del sistema dirigido por la metodología XP en sus fases Exploración y Planificación. Se realiza una propuesta del sistema. Además se presenta la lista de reserva de producto que debe cumplir el dispositivo donde se despliega la herramienta.

2.1 Propuesta de solución

Se confecciona una herramienta de captura de datos para redes TCP/IP llamada Octopus. Dicha herramienta monitoriza el flujo de información entrante y saliente generado por las múltiples aplicaciones y servicios de los teléfonos inteligentes que utilizan Android 2.2. Octopus está basado en tcpdump que es una de las principales herramientas de captura de paquetes. La función principal de Octopus es la captura del tráfico IP que se genera mientras se realizan actividades en la red, tarea que realiza de forma pasiva, sin modificar o transmitir el tráfico de red. En la tarjeta de memoria externa del dispositivo se guarda la información de la captura en un archivo con formato pcap, legible por el WireShark. La captura de paquetes se realiza en tiempo real siempre que el dispositivo esté conectado a una red activa. La herramienta es capaz de mostrar el estado y tipo de red con la cual establece conexión el dispositivo. Como se muestra en la Figura 5.

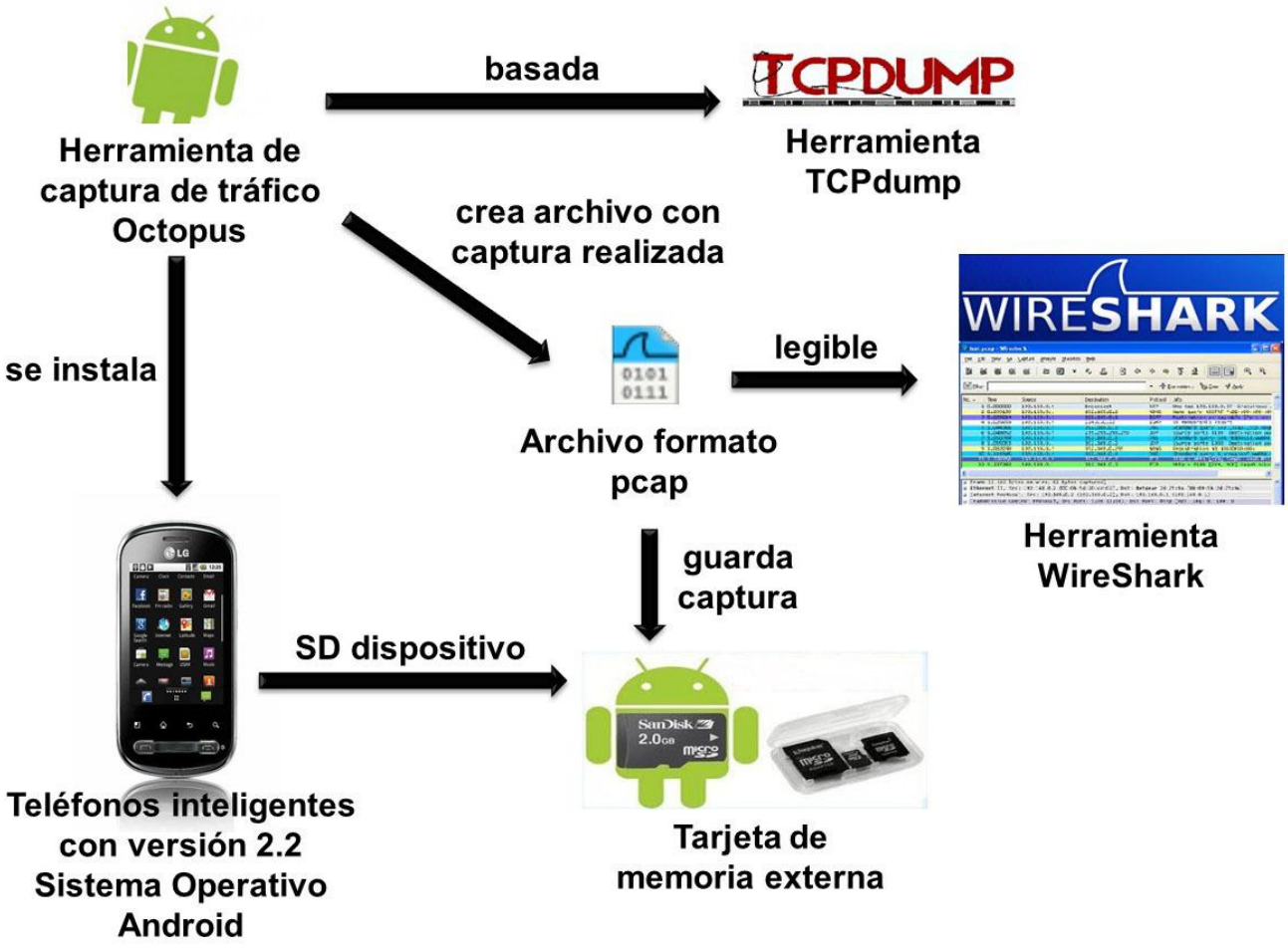


Figura 5: Propuesta del Sistema.

2.2 Persona relacionada con el sistema

Se define como persona relacionada con el sistema al usuario que interactúa directamente con la herramienta. Estos usuarios deben poseer conocimientos de redes como administradores de redes, analistas de redes o especialistas en redes, los cuales son encargados de ejecutar la aplicación según las funcionalidades para la que ha sido creada.

2.3 Funcionalidades del Sistema

Las funcionalidades del sistema son las capacidades o condiciones que el sistema debe cumplir. A continuación se muestran las funcionalidades:

- ✓ Iniciar captura de paquetes.
- ✓ Detener captura de paquetes.

2.4 Lista de reserva del producto

Las listas de reserva del producto en una aplicación son muy importantes ya que son las cualidades que todo sistema debe poseer para un correcto funcionamiento. A continuación se definen los siguientes requisitos:

Usabilidad:

El sistema deberá contar con una interfaz de fácil entendimiento para que usuarios inexpertos puedan interactuar fácilmente con el software.

El sistema podrá ser utilizado por cualquier usuario con las siguientes características:

- ✓ Conocimientos básicos relativos al uso de un teléfono inteligente.
- ✓ Conocimientos básicos del sistema operativo Android.

El sistema se distribuirá en lenguaje español.

Hardware

Para la instalación de la aplicación en un entorno emulado se debe disponer de una computadora de 1GB de RAM o superior, 160 GB de disco duro o superior y un procesador Intel 2.5 GHz. El emulador requiere de la versión 2.2 del sistema operativo Android.

Software

La instalación de la aplicación en un entorno emulado requiere de la máquina virtual de Java en su versión 6, Eclipse Índigo 3.7, SDK de Android 15.0.1 y el plugin ADT 15.0.1. Sistema Operativo Linux la distribución Ubuntu 11.10 o Sistema Operativo Windows. El emulador de Android debe tener permisos de súper-usuarios, la aplicación superuser_3.0.apk y la aplicación SuperSU_1.00.apk.

2.5 Fase de Exploración

El ciclo de vida de un proyecto realizado con la metodología XP inicia con la fase de Exploración. En esta fase, los clientes plantean a grandes rasgos las Historias de Usuarios. Al finalizar el equipo cuenta con suficiente material de trabajo como para producir una primera entrega. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas y tecnologías que serán utilizadas en el proyecto.

2.5.1 Historias de Usuario

Las historias de usuarios (HU) son una breve descripción del comportamiento que posee el sistema a desarrollar. Son escritas por los propios clientes, tal y como quieren ellos que funcione el sistema, no quita que los desarrolladores los puedan ayudar en la identificación de las mismas. Estas emplean la terminología del cliente sin lenguaje técnico. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Las HU se clasifican según:

La prioridad en el negocio:

> **Alta**: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
>
> **Media**: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

Baja: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

El riesgo en su desarrollo:

Alta: Cuando en la implementación de las HU se consideran la posible existencia de errores que conlleven a la inoperatividad del código.

Media: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.

Baja: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Las HU son representadas mediante tablas divididas por las siguientes secciones:

- ✓ Número: esta sección representa el número, incremental en el tiempo, de la historia de usuario que se describe.
- ✓ Nombre de Historia de Usuario: identifica la HU que se describe entre los desarrolladores y el cliente.
- ✓ Modificación de Historia de Usuario Número: sección que representa si la HU se le realizó alguna modificación con respecto al estado anterior.
- ✓ Usuario: Los programadores responsables de la historia de usuario.
- ✓ Iteración asignada: número de la iteración donde va a desarrollarse la HU.
- ✓ Prioridad en negocio: se le otorga una prioridad (Alta, Media, Baja) a las HU de acuerdo a la necesidad de desarrollo.
- ✓ Riesgo en Desarrollo: se le otorga una medida de (Alto, Medio, Bajo), a la ocurrencia de errores en el proceso de desarrollo de la HU.

- ✓ Puntos Estimados: es el tiempo estimado en semanas que se demorará el desarrollo de la HU.
- ✓ Puntos Reales: representa el tiempo que se demoró en realidad el desarrollo de la HU.
- ✓ Descripción: breve descripción de la HU.
- ✓ Observaciones: señalamiento o advertencia del sistema.

A continuación se exponen las HU definidas por el cliente en conjunto con equipo de desarrollo (Tabla 1 y Tabla 2):

Historia de Usuario	
Numero: 1	**Título:** Iniciar captura
Modificación de historia de usuario: Ninguna	
Usuario: Dainé Sánchez Fuentes Alan Beltrán Graverán	**Iteración:**1
Prioridad: Alta	**Puntos Estimados:** 2
Riesgo de desarrollo: Alto	**Puntos Reales**
Descripción: El usuario oprime el botón Iniciar y la herramienta comienza la captura de todos los paquetes que circulan por la red del usuario. Se crea el archivo pcap en la tarjeta SD donde se van guardando los datos. El archivo tiene por nombre “cap” día, mes, año, hora, minuto, segundo en formato de 24 horas.	
Observaciones: El dispositivo debe tener permisos súper-usuario.	

Tabla 1: HU Iniciar captura.

Historia de Usuario	
Numero: 2	**Título:** Detener captura
Modificación de historia de usuario: Ninguna	
Usuario: Dainé Sánchez Fuentes	**Iteración:** 2

Alan Beltrán Graverán	
Prioridad: Alta	**Puntos Estimados:** 2
Riesgo de desarrollo: Alto	**Puntos Reales**
Descripción: Las sesiones de captura de paquetes finalizan cuando el usuario lo decida oprimiendo el botón Detener o cuando la capacidad de la sesión supere los 10MB de contenido.	
Observaciones: Debe haber sido iniciada una sesión de captura previamente.	

Tabla 2: HU Detener captura.

2.6 Fase de planificación

En esta fase el cliente establece la prioridad que tendrá cada HU según sus necesidades más inmediatas, luego los programadores realizan una estimación del esfuerzo que se necesita para cada una de ellas. La fase de planeamiento toma un par de días. Se deben incluir varias iteraciones para lograr un release. El cronograma fijado en la etapa de planeamiento se realiza a un número de iteraciones, cada una toma de una a cuatro semanas en ejecución.

2.6.1 Estimación de esfuerzo

La Estimación por Esfuerzo consiste en asignarle puntos a las HU y cada uno de estos puntos son equivalentes a una semana. Los programadores se basan para realizar la estimación en la complejidad que pueden tener las HU y en el tiempo hábil para el desarrollo de la aplicación. Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos.

Para el logro del sistema se ha realizado la estimación de esfuerzo por cada HU, teniendo en cuenta la complejidad de cada una de las funcionalidades requeridas por el cliente. A continuación se muestran las estimaciones de esfuerzo para cada HU en la tabla 3.

Historia de usuario	Puntos de estimación
Iniciar captura	2
Detener captura	2

Tabla 3: Estimación de esfuerzo por HU.

2.6.2 Plan de Iteraciones

Después de identificadas y descritas las HU y estimar el esfuerzo dedicado a la realización de cada una de ellas, en XP se procede a generar el artefacto plan de iteraciones. El mismo muestra cuales son las HU que serán implementadas en cada iteración del sistema para mejorar el desempeño del equipo de desarrollo. Al final de la última iteración el sistema estará listo para entrar en producción. Tratando de esta manera de tener preparadas las funcionalidades básicas e indispensables del sistema.

Debido al rigor de las funcionalidades se decide realizar el sistema en 2 iteraciones, las cuales se detallan a continuación:

Iteración 1:

En esta iteración se implementa la historia de usuario No1 dando al sistema la primera funcionalidad. Como resultado de la iteración el sistema será capaz de capturar el tráfico que circula por la red a la que el usuario se encuentra conectado y guardar la información obtenida en un archivo con extensión pcap.

Iteración 2:

En esta iteración se implementa la historia de usuario No2 incluyéndole al sistema la posibilidad de finalizar la captura que ha sido iniciada. Como resultado de esta iteración se tendrá una herramienta capaz de capturar el tráfico IP que genera el usuario al realizar sus actividades en la red.

2.6.3 Plan de duración de iteraciones

Como parte del ciclo de vida de un proyecto utilizando la metodología XP se crea el plan de duración de iteraciones. Para la confección del plan se tiene en cuenta la estimación de esfuerzo para cada HU. Este

plan permite mostrar la duración real de cada iteración así como el orden en que se implementarán las HU, teniendo una mayor organización. Como se muestra en la tabla 4.

Iteraciones	Orden de las HU a implementar	Duración Total
1	Iniciar captura	2 semanas y 3 días
2	Detener captura	1 semanas y 2 días

Tabla 4: Plan de duración de iteraciones.

2.6.4 Plan de entregas

Se presenta el Plan de Entrega estimado para la fase de implementación, detallando la fecha de fin para cada una de las iteraciones definidas en la tabla 5.

Iteración	Fecha de entrega
1	3 de mayo del 2013
2	12 de mayo del 2013

Tabla 5: Plan de entregas.

Conclusiones del capítulo

Con la realización de este capítulo se trazan las bases para el desarrollo del sistema. Se plasman las funcionalidades del sistema con detalles específicos de su implementación. Además, se realiza el plan de entregas donde se indican las funcionalidades creadas para cada versión del programa y las fechas en las que se publican estas versiones. Se obtiene una planificación del tiempo de desarrollo de las iteraciones y el orden en que se implementan las funcionalidades de acuerdo con la prioridad que le sea asignada.

Capítulo 3: Diseño del Sistema

Introducción del capítulo

El presente capítulo elabora una propuesta del sistema a desarrollar mediante la creación de los artefactos correspondientes a la fase de diseño según plantea la metodología XP. Se identifican y organizan las clases relevantes para las funcionalidades del sistema, así como los patrones arquitectónicos y de diseño utilizados para la realización de la herramienta.

3.1 Diseño

Durante el diseño de la solución, la máxima simplicidad posible es la clave para el éxito de XP. Se debe tener en cuenta que un diseño complejo siempre tarda más en desarrollarse que uno simple, y que siempre es más fácil añadir complejidad a un diseño simple que quitarla de uno complejo.

XP construye un proceso de diseño evolutivo que se basa en refactorizar un sistema simple en cada iteración. Todo el diseño se centra en la iteración actual y no se hace nada anticipadamente para necesidades futuras. El resultado es un proceso de diseño disciplinado, combina la disciplina con la adaptabilidad de una manera que indiscutiblemente la hace una de las más desarrolladas de entre todas las metodologías ágiles.

3.2 Patrón arquitectónico

Un patrón arquitectónico define la estructura básica de una aplicación, provee un subconjunto de subsistemas predefinidos, incluyendo reglas, lineamientos para conectarlos y pautas para su organización y constituye una plantilla de construcción.

Entre las ventajas del uso de patrones, se pueden encontrar:

- ✓ Permiten la reutilización de soluciones arquitectónicas de calidad.
- ✓ Son de gran ayuda para controlar la complejidad de un diseño.
- ✓ Facilitan la documentación de diseños arquitectónicos.
- ✓ Proporcionan un vocabulario común que mejora la comunicación entre diseñadores.

Los patrones arquitectónicos expresan una organización estructural para un sistema, permiten estructurar los componentes y subsistemas de un sistema. La abstracción más alta en cuanto a soluciones a través de patrones, se obtiene a través del uso de patrones de arquitectura.

3.2.1 Patrón arquitectónico N-Capas

Se propone el uso del patrón arquitectónico N-Capas para el desarrollo de la aplicación, ya que los entornos de ejecución de dicha aplicación estarán distribuidos en diferentes elementos o nodos tanto lógicos como físicos. Aquí aparecen dos conceptos importantes: las capas que se encargan de la distribución lógica de los componentes, y los niveles que se refieren a la colocación física de los recursos. La característica principal de este patrón es que cada capa oculta las capas inferiores de las siguientes superiores a esta (Figura 6).

Algunas de las ventajas de utilizar este patrón son (28):

- ✓ Mejoras en las posibilidades de mantenimiento, debido a que cada capa es independiente de la otra los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo.
- ✓ Escalabilidad, ya que las capas están basadas en diferentes máquinas, el escalamiento de la aplicación hacia afuera es razonablemente sencillo.
- ✓ Flexibilidad, como cada capa puede ser manejada y escalada de forma independiente, la flexibilidad se incrementa.
- ✓ Disponibilidad, las aplicaciones pueden aprovechar la arquitectura modular de los sistemas habilitados usando componentes que escalan fácilmente lo que incrementa la disponibilidad.

La descomposición en capas que se propone es (28):

Capa de presentación: es la única que interactúa directamente con el usuario presentándole el sistema, permitiendo el intercambio de información entre ambos. Contiene una interfaz gráfica que posibilita mostrar a los usuarios los resultados de sus peticiones y estados de la aplicación. Esta capa solo interactúa con la capa de negocio, que es su inferior inmediata.

Capa de negocio: también conocida como lógica de negocio, es la que recibe las peticiones de la capa de presentación y le envía las respuestas tras el proceso. Aquí se realiza la mayor parte del

procesamiento de la información del dominio de la aplicación, donde se ejecutan los algoritmos específicos del programa. Esta capa interactúa con la capa de presentación, para recibir sus solicitudes y presentarle los resultados.

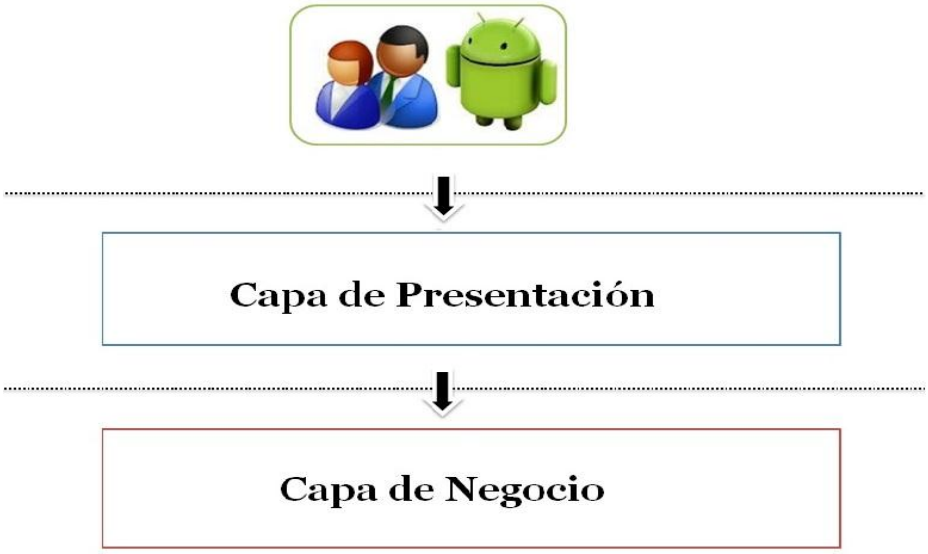


Figura 6: Patrón arquitectónico 2 Capas.

A continuación se describen los elementos de cada capa (Figura 7):

- ✓ Capa de presentación
  - En esta capa se encuentra la actividad Android Octopus que contiene los componentes visuales que se muestran al usuario.
- ✓ Capa de negocio
  - En esta capa se representan todos los componentes lógicos del flujo de trabajo, o sea las clases Javas que se encargan de ejecutar las tareas propias de la aplicación.

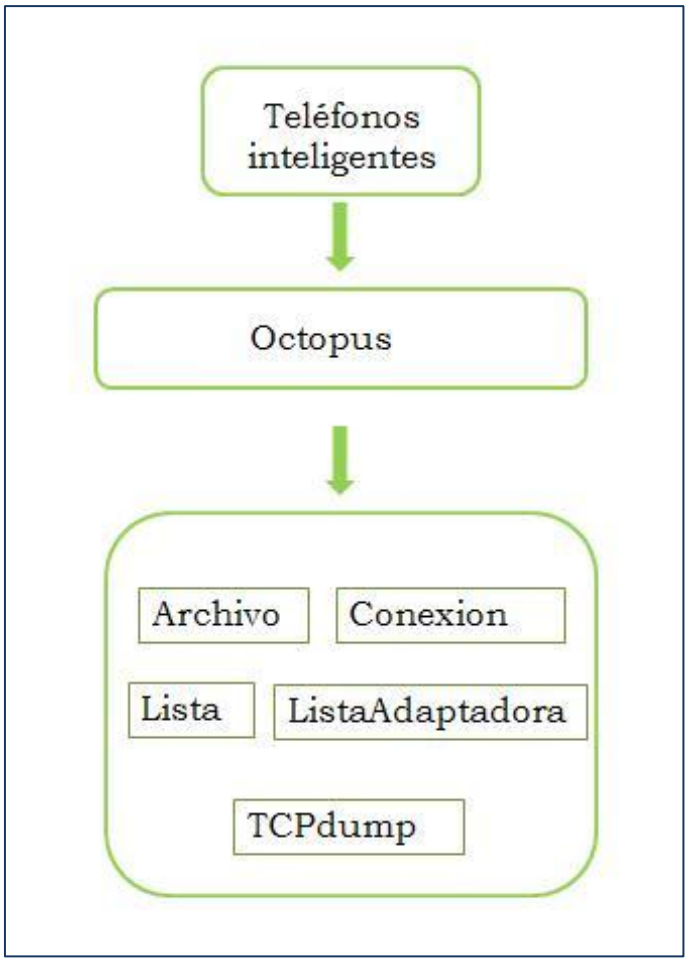


Figura 7: Diagrama orientado a dominio del patrón arquitectónico.

3.3 Patrones de Diseño

Un patrón de diseño constituye un esquema para refinar subsistemas o componentes. Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades, además de que ayuda a construir clases y a estructurar sistemas de clases. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces (29).

Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería de software, sino simplemente tratar de darle una representación a principios ya existentes que quizás ya han sido usado en los proyectos por un mero sentido de la lógica (29).

Un patrón de diseño es (29):

- ✓ Una solución estándar para un problema común de programación.
- ✓ Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- ✓ Un proyecto o estructura de implementación que logra una finalidad determinada.
- ✓ Un lenguaje de programación de alto nivel.
- ✓ Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- ✓ Conexiones entre componentes de programas.
- ✓ La forma de un diagrama de objeto o de un modelo de objeto.

Ventajas (29):

- ✓ Proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifica y describe formas de solucionar problemas que ocurren de forma frecuente en el desarrollo.
- ✓ Están basados en la recopilación del conocimiento de los expertos en desarrollo de software.
- ✓ Es una experiencia real, probada y que funciona.
- ✓ Facilitan la localización de los objetos que formarán el sistema, la determinación de la granularidad adecuada, el aprendizaje y la comunicación entre programadores.
- ✓ Especifican interfaces para las clases e implementaciones al menos parciales.

3.3.1 Patrones GRASP[13]

Los patrones GRASP son patrones generales de software para la asignación de responsabilidades a objetos. Describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades y constituyen la base del cómo se diseñará el sistema. Los patrones GRASP son: Experto, Creador, Bajo Acoplamiento, Alta Cohesión y Controlador (29).

[13] Responsabilidad de la Asignación General de Patrones de Software

Experto

Un Experto es una clase que tiene toda la información necesaria para implementar una responsabilidad. La respuesta es asignar la responsabilidad a la clase que contenga la información necesaria para cumplir la responsabilidad, o sea la clase debe ser la experta en la información. La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). En la aplicación este patrón se evidencia en la clase TCPdump.java.

Creador

Para guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El diseño bien asignado permitirá soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. En la aplicación se utiliza este patrón en la clase TCPdump.java.

Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está relacionada a otras clases. Una clase con bajo o débil acoplamiento no depende de muchas otras. Es la idea de tener las clases 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 reutilización, y disminuyendo la dependencia entre las clases. La solución es asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible, sin comprometer la funcionalidad por supuesto. En la aplicación se evidencia este patrón en las clases Archivo.java y Conexión.java porque no tienen dependencia de ninguna otra clase y en TCPdump.java porque tiene la menor dependencia posible de otras clases.

Alta Cohesión

Cada elemento del diseño debe realizar una labor única dentro del sistema, lo cual expresa que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible

relacionada con la clase. En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión tiene responsabilidades estrechamente relacionadas y poco complejas. Este patrón se refleja en las clases Archivo.java y Conexión.java.

3.3.2 Patrones GoF

El avance reciente más importante en el diseño orientado a objetos es probablemente el movimiento de los patrones de diseño, inicialmente narrado en "Design Patterns (Patrones de Diseño)", por Gamma, Helm, Johnson y Vlissides que suele llamarse el libro de la "Banda de los Cuatro" (en inglés, GoF: Gang of Four). Los patrones de diseño GoF se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento desglosados en 23 patrones (30).

Estructurales: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. De los patrones definidos como estructurales se utiliza el adaptador.

- ✓ Adaptador (Adapter): convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite que cooperen clases que de otra manera no podrían por tener interfaces incompatibles. Dentro de la plataforma móvil, puede emplearse para generar los elementos de componentes visuales como las listas expandible, que requieren de un adaptador para crear los grupos padres y los hijos, que han de ser mostrados al usuario. Este patrón se utiliza en la clase ListaAdaptadora.java.

3.4 Tarjetas CRC (Clases-Responsabilidad-Colaboración)

Las tarjetas CRC proponen una forma de trabajo, preferentemente grupal, para encontrar los objetos del dominio de la aplicación, sus responsabilidades y cómo colaboran con otros para realizar tareas, las cuáles registran el nombre de las clases, sus responsabilidades y las otras clases con la que colaboran (31).

Archivo	
Responsabilidad	**Colaboraciones**
crearNombre	
consultarEstadoDeLaSD	

Tabla 6: Tarjeta CRC Clase Archivo.

Conexion	
Responsabilidad	**Colaboraciones**
comprobarConexion	

Tabla 7: Tarjeta CRC Clase Coneccion.

TCPdump	
Responsabilidad	**Colaboraciones**
iniciarCaptura	Archivo
detenerCaptura	
picar(Stringlinea)	
crearComando	

Tabla 8: Tarjeta CRC Clase TCPdump.

3.4.1 Interfaz de Usuario

La interfaz es la vía que usa el usuario para comunicarse con la aplicación. La interfaz de usuario conocida también como UI del inglés *User Interface* incluye toda clase de componentes visuales que facilitan la interacción usuario/aplicación. La Figura 8 muestra la interfaz de usuario de la herramienta.

Figura 8: Interfaz de Usuario.

Conclusiones del capítulo

En este capítulo se realizó el análisis del sistema, se definieron los patrones arquitectónicos y de diseño usados con el objetivo de lograr una mayor organización en los elementos que conforman la aplicación y se propuso un prototipo de interfaz para la interacción con la herramienta.

Capítulo 4: Implementación y Pruebas

Introducción del capítulo

En el presente capítulo se documentan las fases de implementación y prueba según lo propone la metodología de desarrollo XP. Se desglosan las HU en tareas con el objetivo de facilitar el trabajo de los desarrolladores y se muestran los resultados de las pruebas de aceptación realizadas a la aplicación.

4.1 Implementación

En esta fase XP propone tener en cuenta una serie de aspectos como son la disponibilidad del cliente y el desarrollo en pareja para lograr mayores resultados en la implementación del software.

Disponibilidad del cliente: El cliente formó parte del equipo de desarrollo, describió las HU, guío la toma de decisiones, aprobó las versiones del producto y verificó el cumplimiento de los objetivos trazados.

Desarrollo en pareja: Toda la implementación fue realizada por dos personas que trabajaron en forma conjunta.

4.1.1 Tareas de la Ingeniería

Para llevar a cabo la correcta implementación de las HU se deben definir por parte del equipo de desarrollo las TI (Tareas de Ingeniería) que se realizarán en cada una de las iteraciones. Las TI también conocidas como tareas de implementación permiten a los desarrolladores obtener un nivel de detalle más avanzado que el que propicia las HU.

El desarrollo del software se planificó en dos iteraciones de trabajo. Para la implementación de las HU correspondientes a la primera iteración se definieron las siguientes TI:

Tarea de Ingeniería	
Número de Tarea:1	**Número de la HU:**1
Nombre de Tarea: Iniciar captura	
Tipo de Tarea: Desarrollo	
Fecha Inicio:15/4/2013	**Fecha Fin:**3/5/2013
Programador Responsable: Dainè Sánchez Fuentes	
Descripción: La aplicación comienza a capturar paquetes de la red, instala el binario tcpdump y también guarda la captura conjuntamente en un archivo con formato pcap.	

Tabla 9: Tarea de Ingeniería Iniciar captura iteración 1.

Tarea de Ingeniería	
Número de Tarea:2	**Número de la HU:**2
Nombre de Tarea: Detener captura	
Tipo de Tarea: Desarrollo	
Fecha Inicio:3/5/2013	**Fecha Fin:**12/5/2013
Programador Responsable: Alan Beltrán Graverán	
Descripción: La aplicación finaliza la captura de paquetes.	

Tabla 10: Tarea de Ingeniería Detener captura iteración 2.

4.2 Estándares de nomenclatura y codificación utilizados

Los estándares de codificación son un conjunto de directrices, normas y reglamentos enfocados a la especificación de cómo debe escribirse el código fuente de la aplicación. Estos incluyen pautas sobre la nomenclatura de las variables, clases y paquetes; la correcta identación del código, cómo escribir

estructuras de control, entre otros aspectos. La correcta elaboración y utilización de los estándares de codificación permiten que todos los desarrolladores implementen siguiendo las mismas pautas y así puedan entender el código del resto como si estuvieran mirando el de ellos, así la aplicación será más legible, uniforme y fácil de mantener (32).

4.2.1 Nomenclatura general

- ✓ El nombre de las clases, métodos y variables deben estar escritos en español.
- ✓ El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura lowerCamelCase, que dicta que para un nombre compuesto por varias palabras comenzará con minúscula pero todas las palabras internas que lo componen comienzan con mayúscula.

Paquetes (32)

- ✓ Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales.El paquete base queda definido como com.octopus, en este paquete no se definirá ninguna clase.
- ✓ Se tendrá, así mismo, otro nivel extra dentro del paquete definido como el nombre del proyecto o de la capa.

Identación (32)

- ✓ En el contenido siempre se identará este con tabulaciones, nunca utilizando espacios en blanco.

Clases (32)

- ✓ El nombre de las clases comenzará con mayúsculas y cada salto de palabras debe iniciar con mayúsculas.

Métodos (32)

- ✓ Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (*lowerCamelCase*).

- ✓ No se permiten caracteres especiales.

Nombre de variables

- ✓ El nombre de las variables debe empezar con letra minúsculas y de existir un salto de palabra comenzaría con mayúscula.

4.2.2 Estilos de codificación

Comentarios

- ✓ Como norma es obligatorio proporcionar un comentario de documentación por cada clase y método creado.
- ✓ Comentario de la clase / método:
  - Prescripción genérica de la clase o método y su responsabilidad.
- ✓ Reglas generales a la hora de escribir comentarios de documentación.
  - Siempre se escribe en tercera persona.
  - Los caracteres especiales tales como tildes y eñes se han de codificar con su código HTML correspondiente.
  - Las descripciones siempre deberían empezar por un verbo.

Sentencias

- ✓ Una sentencia por línea de código.
- ✓ Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un if.

4.3 Validación de Propuesta

Uno de los pilares de la metodología utilizada es el proceso de pruebas. XP anima aprobar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar codificaciones y refactorizaciones (27).

XP divide las pruebas del sistema en dos grupos:

- ✓ **Pruebas unitarias**: encargadas de verificar el código y diseñada por los programadores.
- ✓ **Pruebas de aceptación o pruebas funcionales**: destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final.

4.3.1 Pruebas Unitarias

Uno de los métodos utilizados para realizar pruebas de software en la metodología XP son las pruebas unitarias. La base de este método es el hacer pruebas en pequeños fragmentos del código de la aplicación. Estos fragmentos deben ser unidades estructurales del programa encargados de una tarea específica, en programación orientada a objetos se puede afirmar que estas unidades son los métodos o las funciones que se tienen definidos. El objetivo de estas pruebas es el aislamiento de partes del código y la demostración de que no contienen errores. Estas no generan artefactos y no son directamente palpables para el cliente (27).

Resultados de las pruebas unitarias:

Las pruebas unitarias fueron desarrolladas constantemente cada vez que se terminaba de implementar alguna funcionalidad probándola directamente en el entorno real. Dichas pruebas se desarrollan utilizando la extensión Android JUnit, que es una parte integral del SDK de Android. Permite probar componentes específicos mediante clases de casos de pruebas. Estas clases proporcionan métodos auxiliares para la creación de objetos de imitación y métodos que ayudan a controlar el ciclo de vida de un componente (Figura 9).

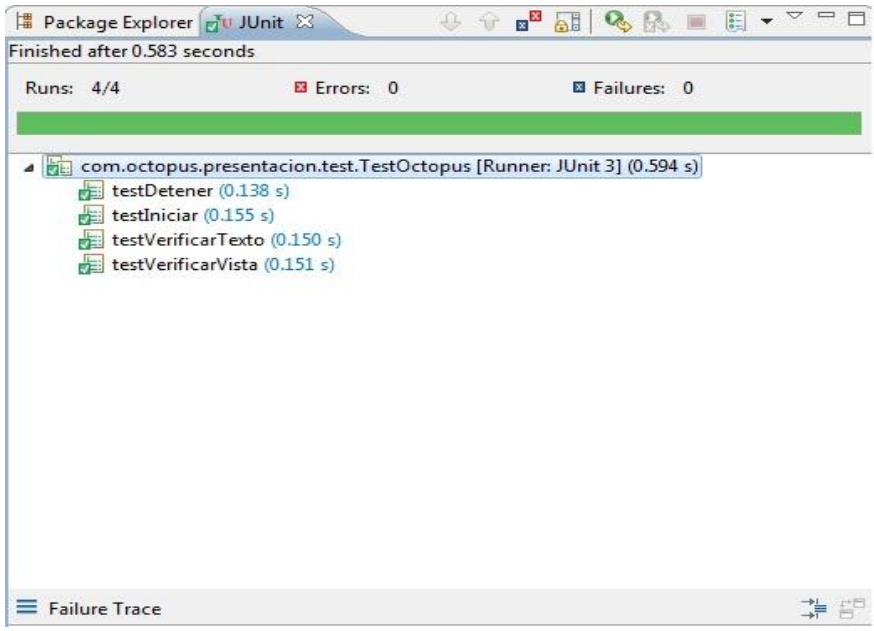


Figura 9: Resultados de las pruebas unitarias.

4.3.2 Pruebas de Aceptación

El objetivo de estas pruebas es verificar los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada (27).

Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera sus pruebas de aceptación. Esto significa que debe desarrollarse un nuevo test de aceptación para cada iteración o se considerará que el equipo de desarrollo no realiza ningún progreso. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas. La garantía de calidad es una parte esencial en el proceso de XP (27).

Las pruebas de aceptación tienen las siguientes funcionalidades (27):

- ✓ **Clases Válidas**: Se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- ✓ **Clases Inválidas:** Se hará la descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado y cómo responde el sistema.
- ✓ **Resultado Esperado**: Se hará una breve descripción del resultado que se espera ya sea para entradas válidas o entradas inválidas.
- ✓ **Resultado de la Prueba**: Se hará una breve descripción del resultado que se obtiene.
- ✓ **Observaciones:** Algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

La realización de este tipo de pruebas y la publicación de los resultados debe ser los más rápido posibles, para que los desarrolladores puedan realizar con la mayor rapidez los cambios que sean necesarios.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario accede a la aplicación y oprime el botón Iniciar.		La aplicación instala el binario tcpdump, crea un archivo en la SD externa en formato pcap el cual contiene la captura de paquetes que se esté realizando en el momento y	Satisfactorio.	El emulador debe tener permisos root y las aplicaciones Superuser_3.0.apk y SuperSU_Pro_v1.00.apk previamente

		muestra una notificación que ya se está capturando.		instaladas para poder trabajar con la aplicación.
	El dispositivo no tiene permisos súper-usuario.	La aplicación muestra una notificación informando que el dispositivo no está rooteado.	Satisfactorio.	

Tabla 11: Prueba de Aceptación #1 para HI Iniciar captura.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario oprime el botón Detener.		El proceso de captura se detiene.	Satisfactorio.	Que tenga una sección de captura iniciada.
	No ha iniciado captura.	Muestra una notificación informando que no ha iniciado captura.	Satisfactorio.	

Tabla 12: Prueba de Aceptación #2 para HI Detener captura.

Resultados de las pruebas de aceptación:

En la etapa de pruebas se realizaron 2 pruebas de aceptación de las cuales arrojaron resultados satisfactorios con un número de 5 no conformidades las cuales fueron resultas con éxito (Figura 10).

Figura 10: Resultados de las Pruebas de Aceptación en cada Iteración.

Conclusiones del capítulo

En este capítulo se llevó a cabo la fase de Implementación y Prueba planteada por la metodología XP. Se realizaron las tareas de ingeniería que dieron solución a todas las HU logrando una mayor organización y rapidez en el desarrollo del sistema y se realizaron las pruebas de aceptación con resultados satisfactorios.

Conclusiones

Luego de realizada la investigación, se pudo arribar a las siguientes conclusiones:

- ✓ El estudio de las principales herramientas de diagnóstico de tráfico IP garantizó una base metodológica sobre los elementos que deben caracterizar el proceso de captura de tráfico IP en dispositivos móviles con Android 2.2.
- ✓ Al evaluarse los procesos de negocio asociados a la captura de tráfico IP se obtuvo un modelo guía para la implementación del sistema mediante la generación de los artefactos correspondientes a los flujos de trabajo propuestos por la metodología XP.
- ✓ La implementación de las funcionalidades de la herramienta, acorde a las pautas de diseño, garantizó el cumplimiento de lo establecido en los objetivos específicos de la investigación.

Recomendaciones

El objetivo general de este trabajo fue alcanzado, pero durante su desarrollo surgieron varias ideas que serían recomendables tener en cuenta para su futuro perfeccionamiento:

- ✓ Mostrar al usuario el contenido de la captura a través de un archivo de texto plano en formato legible para permitir su análisis.
- ✓ Incluir parámetros de captura para que la aplicación pueda capturar de acuerdo a filtros establecidos por el usuario.
- ✓ Exportar el archivo creado hacia una dirección en la red establecida por el usuario.
- ✓ Mostrar más información de las redes.

Referencia Bibliográfica

1. **Luna, Ricardo Riuz.** *Diseño e Instalación de redes inalámbricas y mapas de cobertura.* Estado de Puebla : Universidad Popular autónoma, 2005.

2. *Evaluación de tráfico de voz y datos en las redes celulares.* **Carolina, Rosa García y Rojas, Luis Cañedo.** 2, Venezuela : Universidad Rafael Belloso Chacin: Revista TELEMATIQUE, 2006, Vol. 5. ISSN:1856-4194.

3. **Tanenbaum, Andrew S.** *Computer Networks.* México : Prentice Hall, 2003. 0130661023.

4. **Paratti, Gustavo Gabriel.** *Redes.La guía de referencia actual y definitiva.* Buenos Aires : MP Ediciones Corporation, 2004. ISBN:9789875262089.

5. **E.Comer, Douglas.** *Redes Globales de información con internet y TCP/IP.* México : PRENTICE-HALL HISPANOAMERICANA, S. A., 1996. ISBN:968-880-541-6.

6. **Herrera Pérez, Enrique.** *Tecnologías y redes de transmisión de datos.* México : Limusa, 2003. ISBN:9681863836.

7. **Braden, R.** *RFC: 1122, Requirements for Internet Hosts -- Communication Layers.* October 1989.

8. **Plummer, David C.** *RFC: 826 Un Protocolo Para la Resolución de Dirección Ethernet.* Noviembre 1982.

9. **Postel, J.** *RFC: 792, PROTOCOLO DE MENSAJES DE CONTROL INTERNET (ICMP : INTERNET CONTROL MESSAGE PROTOCOL).* s.l. : Network Working Group, Septiembre 1981.

10. —. *RFC: 793, PROTOCOLO DE CONTROL DE TRANSMISIÓN.* s.l. : Editorial USC/Information Sciences Institute, Septiembre de 1981.

11. —. *RFC: 768, PROTOCOLO DE DATAGRAMAS DE USUARIO.* s.l. : Editorial USC/Information Sciences Institute, 28 Agosto 1980.

12. **J. Reynolds, J. Postel.** *RFC: 854, ESPECIFICACIÓN DEL PROTOCOLO TELNET.* California : USC/Instituto de Ciencias de la Información, mayo 1983.

13. **Postel, J.** *RFC: 959, PROTOCOLO DE TRANSFERENCIA DE FICHEROS.* Octubre 1985.

14. **Postel, Jonathan B.** *RFC: 821, Protocolo simple de transferencia de correo.* California : USC/Instituto de Ciencias de la Información, Agosto 1982.

15. **R. Fielding, J. Gettys, J. Mogul,T. Berners-Lee y otros.** *RFC: 2616, Hypertext Transfer Protocol -- HTTP/1.1.* s.l. : The internet Society, Junio de 1999.

16. **Information Sciences Institute.** *RFC:791 " INTERNET PROTOCOL".* California : University of Southern California, 1981. ISBN:90291.

17. **Silberschatz, Abraham, Baer Galvin, Peter y Gagne, Greg.** *Fundamentos de sistemas operativos.* España : McGraw-Hill, 2006. ISBN:84-481-4641-7.

18. **Stallings, William.** *Sistemas Operativos, 2ed.* Madrid : P R E N T I C E H A L L., 1997. ISBN: 84-89660-22-0.

19. **Tomás Gironés, Jesús.** *El Gran Libro de Android.* Barcelona : Marcombo, 2011. ISBN:8426717322.

20. **Díaz, Almudena, Merino, Pedro y Rivas Tocado, F. Javier.** *SymPA: un Analizador de Protocolos para Dispositivos.* Málaga : Universidad de Málaga. ISNN:29071.

21. **Álvarez, Andrés, y otros, y otros.** *Field measurements of mobile services with Android smartphones.* España : Consumer Communications and Networking Conference (CCNC), IEEE, 2012. ISBN: 978-1-4577-2070-3.

22. **IBM.** Eclipse Platform Technical Overview. Eclipse. [En línea] IBM, 2007. [Citado el: 10 de Enero de 2013.] http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf..

23. **Groussard, Thierry.** *Recursos Informaticos: Java 6.* Barcelona : Editions ENI, 2009. ISSN: 1629-7458.

24. **Ramnath, Rajiv.** *Android 3 SDK Programming for Dummies.* New Jersey : John Wiley & Sons,Inc., 2011. ISBN 978-1-118-00825-6.

25. developer.android.com. [En línea] Febrero de 2013. [Citado el: 15 de Marzo de 2013.] http://developer.android.com/tools/sdk/eclipse-adt.html.

26. **Northcutt, Stephan, Novak, Judy.** *Network Instrusion detection.* s.l. : Third Edition, September 2002. ISBN: 0-73571-265-4.

27. **Rodríguez Corbea, Maite, Ordóñez Pérez, Meylin.** *La Metodología XP Aplicable al Desarrollo del Software Educativo en Cuba.* Universidad de las Ciencias Informáticas, Cuba : s.n., 2007.

28. **Ferrás, Claudia Beatriz Larramendi.** *Propuesta de arquitectura para desarrollo de aplicaciones compuestas para dispositivos móviles con Android.* La Habana : s.n., 2012.

29. **Erich Gamma, R.H., Ralph Johnson, John VLissides.** *Patrones de Diseño.Elementos de software orientado a objetos reutilizables.* España : s.n., 2003.

30. **Debrauwe, Laurent.** *Patrones de diseño para C# - Los 23 modelos de diseño: descripción y soluciones ilustradas en UML 2 y C#.* Barcelona : Editions ENI, Febrero 2012. ISBN: 978-2-7460-7260-2.

31. Actividades de Implementacion Tarjetas CRC. [En línea] [Citado el: 5 de mayo de 2013.] http://www.inf.utfsm.cl/~visconti/xp/Tarjetas_CRC_2.doc.

32. **Vermeulen, Alan y W. Ambler, Scott.** *The Elements of Java(TM) Style.* New York : Cambridge University Press, 2001. ISBN:0521777682.

Bibliografía

1. **Luna, Ricardo Riuz.** *Diseño e Instalación de redes inalámbricas y mapas de cobertura.* Estado de Puebla : Universidad Popular autónoma, 2005.

2. *Evaluación de tráfico de voz y datos en las redes celulares.* **Carolina, Rosa García y Rojas, Luis Cañedo.** 2, Venezuela : Universidad Rafael Belloso Chacin: Revista TELEMATIQUE, 2006, Vol. 5. ISSN:1856-4194.

3. **Tanenbaum, Andrew S.** *Computer Networks.* México : Prentice Hall, 2003. 0130661023.

4. **Paratti, Gustavo Gabriel.** *Redes.La guía de referencia actual y definitiva.* Buenos Aires : MP Ediciones Corporation, 2004. ISBN:9789875262089.

5. **E.Comer, Douglas.** *Redes Globales de información con internet y TCP/IP.* México : PRENTICE-HALL HISPANOAMERICANA, S. A., 1996. ISBN:968-880-541-6.

6. **Herrera Pérez, Enrique.** *Tecnologías y redes de transmisión de datos.* México : Limusa, 2003. ISBN:9681863836.

7. **Braden, R.** *RFC: 1122, Requirements for Internet Hosts -- Communication Layers.* October 1989.

8. **Plummer, David C.** *RFC: 826 Un Protocolo Para la Resolución de Dirección Ethernet.* Noviembre 1982.

9. **Postel, J.** *RFC: 792, PROTOCOLO DE MENSAJES DE CONTROL INTERNET (ICMP : INTERNET CONTROL MESSAGE PROTOCOL).* s.l. : Network Working Group, Septiembre 1981.

10. —. *RFC: 793, PROTOCOLO DE CONTROL DE TRANSMISIÓN.* s.l. : Editorial USC/Information Sciences Institute, Septiembre de 1981.

11. —. *RFC: 768, PROTOCOLO DE DATAGRAMAS DE USUARIO.* s.l. : Editorial USC/Information Sciences Institute, 28 Agosto 1980.

12. **J. Reynolds, J. Postel.** *RFC: 854, ESPECIFICACIÓN DEL PROTOCOLO TELNET.* California : USC/Instituto de Ciencias de la Información, mayo 1983.

13. **Postel, J.** *RFC: 959, PROTOCOLO DE TRANSFERENCIA DE FICHEROS.* Octubre 1985.

14. **Postel, Jonathan B.** *RFC: 821, Protocolo simple de transferencia de correo.* California : USC/Instituto de Ciencias de la Información, Agosto 1982.

15. **R. Fielding, J. Gettys, J. Mogul,T. Berners-Lee y otros.** *RFC: 2616, Hypertext Transfer Protocol -- HTTP/1.1.* s.l. : The internet Society, Junio de 1999.

16. **Information Sciences Institute.** *RFC:791 " INTERNET PROTOCOL".* California : University of Southern California, 1981. ISBN:90291.

17. **Silberschatz, Abraham, Baer Galvin, Peter y Gagne, Greg.** *Fundamentos de sistemas operativos.* España : McGraw-Hill, 2006. ISBN:84-481-4641-7.

18. **Stallings, William.** *Sistemas Operativos, 2ed.* Madrid : P R E N T I C E H A L L., 1997. ISBN: 84-89660-22-0.

19. **Tomás Gironés, Jesús.** *El Gran Libro de Android.* Barcelona : Marcombo, 2011. ISBN:8426717322.

20. **Díaz, Almudena, Merino, Pedro y Rivas Tocado, F. Javier.** *SymPA: un Analizador de Protocolos para Dispositivos.* Málaga : Universidad de Málaga. ISNN:29071.

21. **Álvarez, Andrés, y otros, y otros.** *Field measurements of mobile services with Android smartphones.* España : Consumer Communications and Networking Conference (CCNC), IEEE, 2012. ISBN: 978-1-4577-2070-3.

22. **IBM.** Eclipse Platform Technical Overview. Eclipse. [En línea] IBM, 2007. [Citado el: 10 de Enero de 2013.] http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf..

23. **Groussard, Thierry.** *Recursos Informaticos: Java 6.* Barcelona : Editions ENI, 2009. ISSN: 1629-7458.

24. **Ramnath, Rajiv.** *Android 3 SDK Programming for Dummies.* New Jersey : John Wiley & Sons,Inc., 2011. ISBN 978-1-118-00825-6.

25. developer.android.com. [En línea] Febrero de 2013. [Citado el: 15 de Marzo de 2013.] http://developer.android.com/tools/sdk/eclipse-adt.html.

26. **Northcutt, Stephan, Novak, Judy.** *Network Instrusion detection.* s.l. : Third Edition, September 2002. ISBN: 0-73571-265-4.

27. **Rodríguez Corbea, Maite, Ordóñez Pérez, Meylin.** *La Metodología XP Aplicable al Desarrollo del Software Educativo en Cuba.* Universidad de las Ciencias Informáticas, Cuba : s.n., 2007.

28. **Ferrás, Claudia Beatriz Larramendi.** *Propuesta de arquitectura para desarrollo de aplicaciones compuestas para dispositivos móviles con Android.* La Habana : s.n., 2012.

29. **Erich Gamma, R.H., Ralph Johnson, John VLissides.** *Patrones de Diseño.Elementos de software orientado a objetos reutilizables.* España : s.n., 2003.

30. **Debrauwe, Laurent.** *Patrones de diseño para C# - Los 23 modelos de diseño: descripción y soluciones ilustradas en UML 2 y C#.* Barcelona : Editions ENI, Febrero 2012. ISBN: 978-2-7460-7260-2.

31. Actividades de Implementacion Tarjetas CRC. [En línea] [Citado el: 5 de mayo de 2013.] http://www.inf.utfsm.cl/~visconti/xp/Tarjetas_CRC_2.doc.

32. **Vermeulen, Alan y W. Ambler, Scott.** *The Elements of Java(TM) Style.* New York : Cambridge University Press, 2001. ISBN:0521777682.

33. **Romero Hernández, Omar, Muñoz Negrón, David y Romero Hernández, Sergio.** *Introduccion a la ingenieria: Un enfoque industrial.* s.l. : Cengage Learning Editores, 2006.

34. **Sallent Roig, Oriol, Valenzuela González, José Luis y Agustí Comes, Ramón.** *Principios de comunicaciones móviles.* Barcelona : Univiverisdad Politècnica de Catalunya, 2003. ISB: 8483017156.

35. **Nuaymi, Loutfi.** *WiMAX: Technology for Broadband Wireless Access.* Inglaterra : John Wiley & Sons Ltd, 2007. ISBN 978-0-470-02808-7.

36. *Telefonía móvil digital GSM.* **Domínguez Sánchez, Juan José.** Fasc.1, España : Anales de mecánica y electricidad, 2000, Vol. 77. ISSN 0003-2506.

37. *UMTS(Universal Mobile Telecomunication System): El momento de la realidad de tercera generacón.* **Vélez Varela, Fernando.** No.2, Colombia : Entramado, 2007, Vol. 3. ISSN-e:1900-3803.

38. **Holma, Harri, Toskala, Antti.** *LTE-Advanced. 3GPP Solution for IMT-Advanced.* s.l. : Jhon Wiley & Sons, 2012. ISBN 978-1-119-97405-5.

39. **3GPP, TS 23.401.** http://www.etsi.org. [En línea] Junio de 2011. [Citado el: 5 de Diciembre de 2012.] http://www.etsi.org/deliver/etsi_ts/123400_123499/123401/08.14.00_60/ts_123401v081400p.pdf.

40. **Cisco.** http://www.cisco.com. [En línea] 6 de Febrero de 2013. [Citado el: 16 de Marzo de 2013.] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.

41. **Goncalves, Antonio.** *Java EE 5.* Paris : Editions Eyrolles, 2011. ISBN : 978-2-212-12658.

42. **Lee, Wei-Meng.** *Beginning Android Application Development .* Indiana : Wiley Publishing, Inc., 2011. ISBN: 978-1-118-01711-1.

43. **Darwin, Ian.** *Android Cookbook.* Estados Unidos : O'Reilly Media, Inc, 2011 . ISBN: 978-1-449-38841-6.

44. **James Steele, Nelson To.** *The Android developer's cookbook : building applications with the Android SDK.* Boston : Pearson Education, Inc., 2011. ISBN-10: 0-321-74123-4.

45. **Murphy, Mark L.** *The Busy Coder's Guide to Android Development.* Estados Unidos : CommonsWare, LLC., 2008. ISBN: 978-0-9816780-0-9.

46. **Komatineni, Sayed Y. Hashimi and Satya.** *Pro Android.* Estados Unidos : Apress,Inc., 2009. ISBN-13 (pbk): 978-1-4302-1596-7.

47. **Burnette, Ed.** *Hello, Android Introducing Google's Mobile Development Platform, 3rd Edition.* Texas : Pragmatic Programmers, LLC., 2010. ISBN-10: 1-934356-56-5.

48. **Donn Felker, Joshua Dobbs.** *Android Application Development for Dummies.* Indiana : Wiley Publishing, Inc., 2011. ISBN: 978-0-470-77018-4.

49. **Gargenta, Marko.** *Learning Android.* Estados Unidos : O'Reilly Media, Inc., 2011. ISBN: 978-1-449-39050-1.