FACULTAD 2

TÍTULO: DESCUBRIMIENTO DE LA RED V 1.1

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

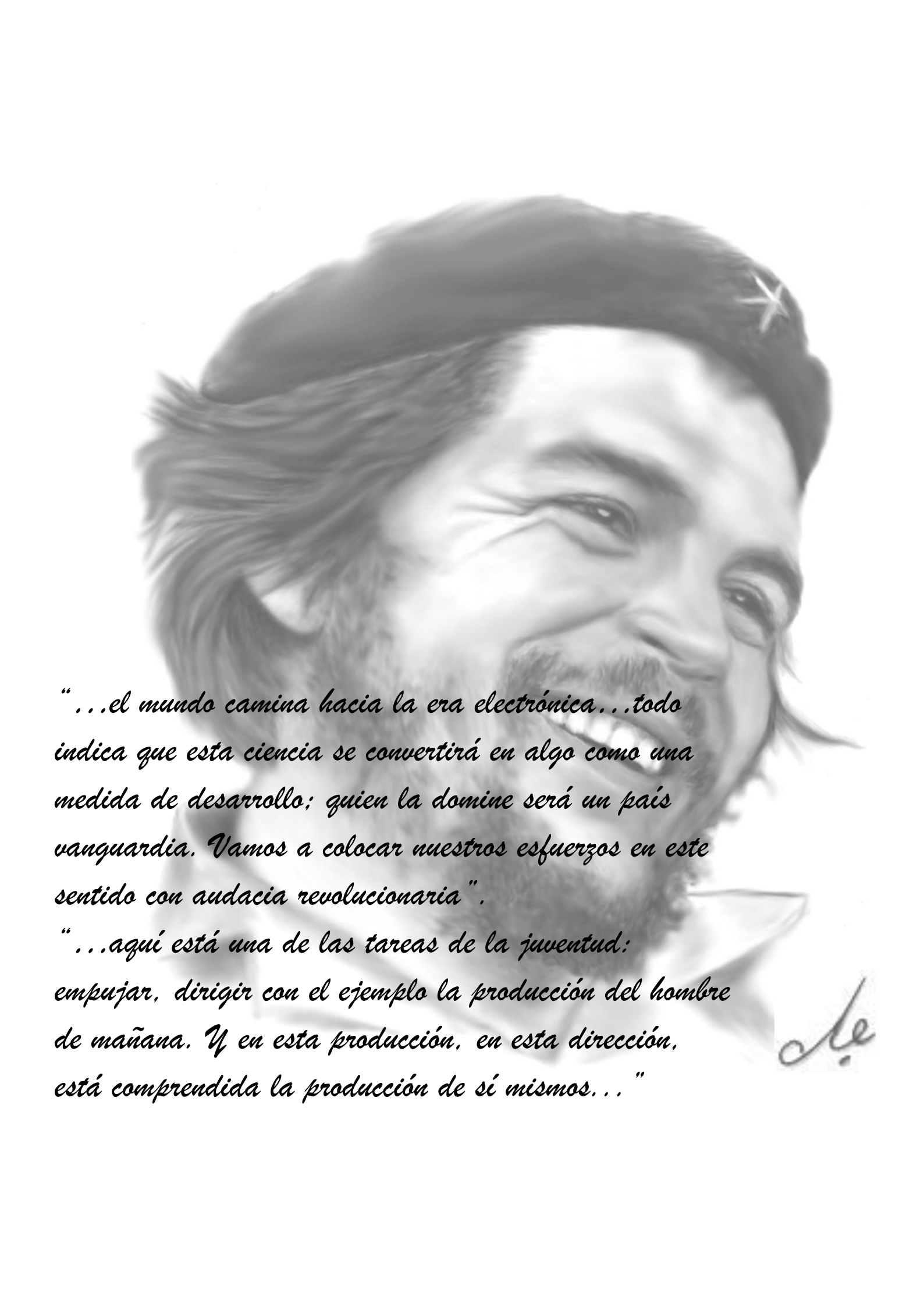
Autor(es): Daniel Martínez Parra

Rafael González Arrieta

Tutor(es): Ing. Lianet Salazar Labrada

Ing. Adrián Ezequiel Mena Rodríguez

La Habana, 2016



"...el mundo camina hacia la era electrónica...todo indica que esta ciencia se convertirá en algo como una medida de desarrollo; quien la domine será un país vanguardia. Vamos a colocar nuestros esfuerzos en este sentido con audacia revolucionaria".

"...aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos..."

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Autores:

_____ _____

Rafael González Arrieta Daniel Martínez Parra

Tutores:

_____ _____

Ing. Adrián E. Mena Rodríguez Ing. Lianet Salazar Labrada

Agradecimientos

A mis padres por confiar más en mi de lo que yo mismo lo hago y que se que me apoyan en cada paso que doy.

A mi familia que todo el tiempo estuvo pendiente de mi día a día

A las amistades que pude hacer en estos cinco años, en especial a mis hermanos de causa: Carlos, el Yoyo, Eric, el Buty, el Jose, Ernestato, la Jota, el Tony y el Chino la Rosa.

A los profesores y tutores que me hicieron pensar como universitario.

A todo aquel que de alguna manera influyó en el transcurso de mi estancia en la universidad, los mejores 5 años de mi vida.

Rafael González Arrieta

A mi familia que siempre me dieron su apoyo, en especial a mi madre por confiar en mi y en cada decisión que he tomado, a mi tíos que siempre me ayudaron en todo lo que necesité y a mis abuelos.

A mi compañero de tesis por creer que siempre pudimos lograrlo a pesar de que somos un poco regados.

A todas las amistades, en especial mi gente del apartamento que siempre me ayudaron cuando lo necesite y por aguantarme los años que vivimos juntos y aquellas otras personas que de alguna forma compartimos buenos momentos.

A los tutores que a pesar de su desconfianza en nosotros desde un inicio siempre demostraron su preocupación.

Daniel Martínez Parra

Resumen

Los avances tecnológicos sobre la informática y las comunicaciones dieron paso al surgimiento y con el paso de los años, al desarrollo de las redes de computadoras. Mediante el uso de las redes de computadoras los usuarios pueden intercambiar información entre sí, además de controlar los recursos de las PC conectadas entre ellas. Con el fin de mantener el control eficaz de los recursos de hardware y software, han surgido innumerables aplicaciones informáticas con el objetivo de facilitar el trabajo a sus usuarios. El Centro Telemática (TLM) de la Universidad de las Ciencias Informáticas (UCI), desarrolló la herramienta Gestor de Recursos de Hardware y Software (GRHS). GRHS posee un módulo que se encarga del descubrimiento de los equipos de cómputo de una red. Este módulo presenta un conjunto de deficiencias que limitan su uso. Por esta razón se decide reconstruir el módulo de descubrimiento de una red de computadoras del sistema GRHS, reutilizando el desarrollo de la versión anterior. Se realizó un estudio del sistema actual para entender el funcionamiento del módulo, el estudio del estado del arte permitió confirmar la utilización de la biblioteca Python-nmap en conjunto con el lenguaje de programación Python 2.7 y basándose en la metodología de desarrollo Programación Extrema (XP). Se obtuvo un nuevo módulo que elimina las deficiencias de su predecesor y permite su empleo en cualquier institución donde sea desplegado GRHS.

Palabras clave: módulo, Python-nmap, escaneo, usuario, subredes, GRHS

Índice

Índice de tablas

Índice de figuras

Introducción

Una red de computadoras es un sistema donde los elementos que la integran son autónomos y están conectados entre ellos a través de medios ya sean físicos o lógicos y pueden comunicarse para compartir recursos, dichos recursos se componen por periféricos, carpetas o documentos. Los primeros enlaces entre ordenadores se caracterizaron por realizarse entre computadoras que utilizaban sistemas operativos idénticos con igual condición de hardware y empleaban líneas de transmisión exclusivas para enlazar solo dos elementos de la red. El surgimiento de las redes de computadoras no solo significó una revolución en el campo de las tecnologías de la comunicación, sino también en la noción que hasta ese momento se tenía de la ciencia de la computación. La conexión de diferentes computadoras entre sí, supuso una novedad, pues hasta ese entonces estaban aisladas en su funcionamiento. Ello posibilitó desde un inicio, intercambiar los recursos de computación e información dentro de las propias universidades y centros de investigación y, posteriormente, con cualquier otra entidad conectada. Conseguir la comunicación entre sí de distintos computadores, figuró un gran brío de investigación para dar solución a los distintos inconvenientes asociados: capacidad de la comunicación, fiabilidad y control de errores o gestión de los medios físicos a través de los cuales realizar la comunicación por solo citar algunas de estas problemáticas.

Cuba no se encuentra ajena a los avances tecnológicos sobre la informática y las comunicaciones, con el objetivo de impulsar el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) creó la Universidad de las Ciencias Informáticas (UCI) que tiene dentro de sus misiones formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática. (1)

Actualmente en la Facultad 2 de la UCI se encuentra el Centro de Desarrollo de Telemática (TLM), el cual se dedica a desarrollar sistemas y brindar servicios informáticos en las ramas de las Telecomunicaciones y la Seguridad Informática. TLM cuenta con el proyecto Gestor de Recursos de Hardware y Software (GRHS), esta es una herramienta informática que permite gestionar la información de las PC (Computadoras Personales) conectadas a una red de computadoras que han sido inventariadas. Brindando información sobre los componentes físicos y el software que poseen, además de sus localizaciones.

Con el fin de realizar la tarea de inventario de hardware y software de las PC conectadas a la red, este sistema utiliza un servicio instalado en cada PC (Gclient, Cliente de GRHS), el cual se encarga de realizar el inventario antes mencionado y actualizar la información en la base de datos del servidor de GRHS. La instalación de Gclient es realizada por parte de los administradores de tecnología de cada área a la que pertenece. Además, el sistema GRHS actualmente posee un módulo que permite conocer el total de computadoras que se encuentran en una subred determinada, a través del escaneo de la subred en cuestión. Reduciendo la probabilidad de cometer errores por parte de los administradores de tecnología que están designados a esta tarea, pues en ocasiones olvidan algunas áreas de trabajo que deben ser inventariadas. Este módulo permite controlar de manera automática si fue instalado el Gclient en cada una de estas PC. El administrador del sistema GRHS tiene acceso a Gadmin, aplicación web que permite acceder a la información obtenida a través de Gclient, además de monitorear en tiempo real los resultados obtenidos durante el proceso de escaneo de las subredes.

Todas las funciones expuestas anteriormente son realizadas a través del servidor de GRHS (Gserver) y teniendo en cuenta las características del módulo de escaneo, representa una pérdida de tiempo perceptible, pues no cuenta con la simultaneidad del escaneo, permitiéndole escanear más de una subred al mismo tiempo. El módulo actual hace un escaneo de la red enviando peticiones TCP-IP (Protocolo de control de transmisión/Protocolo de Internet) hacia cada uno de los nodos en la red por un puerto determinado, si logra conectarse al nodo por ese puerto, asume que la PC tiene el Gclient instalado, esto produce falsos positivos pues el puerto en cuestión puede estar siendo usado por otra aplicación y no necesariamente Gclient. El módulo realiza el escaneo de la red utilizando una máscara de subred[1] fija, la cual es configurada en la librería Nmap (Network Mapper o Mapeador de Red), limitando al módulo a funcionar correctamente en la red UCI, pues una vez que sea instalado en alguna otra entidad fuera de la universidad no obtendría los resultados esperados.

A partir de la situación problemática anteriormente descrita se define como **problema a resolver:** El proceso de descubrimiento de una red de computadoras en GRHS presenta deficiencias que impiden el aprovechamiento de las potencialidades del módulo.

[1] La máscara de red o redes es una combinación de bits que sirve para delimitar el ámbito de una red de computadoras

Teniendo en cuenta lo anterior el **objeto de estudio** del presente trabajo es: las redes de computadoras y su **campo de acción:** el descubrimiento de una red de computadoras en la UCI. Para dar solución al problema planteado se definió como **objetivo general:** Reconstruir el módulo de descubrimiento de una red de computadoras del sistema GRHS, reutilizando el desarrollo de la versión anterior.

Para cumplir dicho objetivo y dar solución al problema antes planteado se definen las siguientes **Tareas de Investigación:**

- ✓ Análisis de las librerías a utilizar para el desarrollo del módulo.
- ✓ Análisis de sistemas similares que realizan el descubrimiento en una red de computadoras para comprender su funcionamiento.
- ✓ Análisis de la metodología seleccionada y las herramientas definidas por el proyecto GRHS para el desarrollo del módulo.
- ✓ Estudio del funcionamiento del módulo existente para determinar los cambios a realizar en el mismo, teniendo en cuenta que se busca escanear multiples subredes, sin elevar demasiado el uso de recursos de hardware y el tiempo.
- ✓ Estudio del funcionamiento del sistema GRHS para propiciar mayor interoperabilidad entre el sistema y el módulo.
- ✓ Validar la solución propuesta mediante las pruebas de software para darle cumplimiento al objetivo de la investigación.

Desde el punto de vista metodológico fueron empleados durante la investigación los siguientes **métodos científicos:**

Métodos teóricos

Analítico-sintético: este método fue utilizado en el estudio y análisis de la teoría e información, con el objetivo de seleccionar los elementos fundamentales enmarcados en el objeto de estudio y campo de acción.

Histórico-lógico: se realizó el estudio de la primera edición del módulo creado para el sistema GRHS, obteniendo una mayor comprensión del funcionamiento del mismo para su futura evolución.

Métodos Empíricos

Observación: Utilizado para detectar las necesidades existentes y posibles mejoras que se le pueden añadir al sistema en desarrollo, con el propósito de obtener un módulo mejor estructurado.

Modelación: Mediante este método se realiza una abstracción con vistas a explicar la realidad. Este método se utilizará para la modelación de los diagramas correspondientes a la fase de análisis y diseño.

El presente trabajo se estructura en tres capítulos:

Capítulo 1: Fundamentación Teórica. Se precisan elementos teóricos que sustentan la investigación y el desarrollo del tema propuesto, a través del estudio y análisis de soluciones existentes. Se describen las herramientas, tecnologías y la metodología a utilizar en el desarrollo de la solución propuesta.

Capítulo 2: Exploración y Planificación. Realiza una descripción de la propuesta de solución y sus funcionalidades, se especifican las Historias de Usuario (HU) definidas por cada una de las funcionalidades, además de las principales características que irá a ostentar el módulo a evolucionar.

Capítulo 3: Diseño, Implementación y Pruebas. Se describen las tareas a realizar para llevar a cabo la implementación de las principales funcionalidades definidas para la evolución del módulo. Se muestra el análisis realizado sobre los tipos de pruebas que se pueden aplicar para validar el funcionamiento de la propuesta de solución. Se describe el proceso de aplicación del tipo de prueba seleccionada y se muestran los resultados obtenidos.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

Teniendo en cuenta los factores que pueden incidir en la implementación del módulo a desarrollar, es necesario evaluar tres puntos fundamentales para la comprensión del estudio que se realiza. Entre los citados se encuentran las bibliotecas usadas para el descubrimiento de red, los protocolos de referencia estudiados para la investigación y sistemas que utilizan estos métodos de descubrimiento de red que se usan actualmente, así como la metodología de desarrollo empleada y herramientas a usar conforman este primer capítulo.

1.2 Bibliotecas utilizadas para realizar el descubrimiento de la red de computadoras

1.2.1 Nmap

Nmap es una herramienta de código abierto empleado para exploración de red y auditoría de seguridad. Se diseñó para analizar rápidamente grandes redes, aunque funciona muy bien contra equipos individuales. Nmap utiliza paquetes IP en formas originales para determinar qué equipos se encuentran disponibles en una red, qué servicios (nombre y versión de la aplicación) ofrecen, qué sistemas operativos (y sus versiones) ejecutan, qué tipo de filtros de paquetes o cortafuegos se están utilizando. Aunque generalmente se utiliza Nmap en auditorías de seguridad, muchos administradores de redes y sistemas lo utilizan para realizar tareas rutinarias, como puede ser el inventariado de la red, la planificación de actualización de servicios y la monitorización del tiempo que los equipos o servicios se mantiene activos. (2)

1.2.2 NTOP

NTOP (Network TOP) es una herramienta que permite monitorizar en tiempo real los usuarios y aplicaciones que están consumiendo recursos de red en un instante concreto y además es capaz de ayudar a la hora de detectar malas configuraciones de algún equipo. Una de las cosas buenas de esta herramienta es que se puede utilizar un navegador web para gestionar y navegar a través de la información de tráfico NTOP para comprender mejor el estado de la red. (3)

1.2.3 Python-nmap

Python-nmap es una librería que se encuentra en desarrollo continuo. Soporta todas las características disponibles en la última versión de Nmap. Existen dos versiones que permiten operar con Python 3.x y Python 2.x, lo cual es importante tener en cuenta a la hora de instalar la librería. Por otro lado, con esta librería es posible ejecutar escaneos de forma síncrona o asíncrona. Un escaneo síncrono, consiste en que python-nmap bloquea el hilo de ejecución principal hasta que el escaneo finalice su ejecución. Un escaneo asíncrono ejecuta el escaneo Nmap en un nuevo hilo de ejecución y permite que el hilo principal siga ejecutándose. En un escaneo asíncrono, se permite la definición de una función de callback que puede ser invocada de forma automática cuando se obtenga un resultado sobre alguno de los puertos analizados. (4)

1.3 Protocolos de referencia en la investigación

1.3.1 Protocolo RIP

El protocolo RIP (Routing Protocol Information o Protocolo de encadenamiento de información), es uno de los protocolos de enrutamiento interior más sencillos y con mayor uso. RIP es un protocolo de versión de distancia de tipo estándar, de puerta de enlace interna o IGP (Internal Gateway Protocol) utilizado por los routers, aunque también pueden actuar en equipos, para intercambiar información acerca de redes IP. (5)

1.3.2 Protocolo OSPF

El protocolo Open Shortest Path First (OSPF) es un Protocolo de gateway interior que se usa para distribuir información de enrutamiento dentro de un sistema autónomo único. OSPF utiliza la multidifusión IP para enviar actualizaciones de estado de enlace. Esto garantiza un menor procesamiento en los routers que no están a la escucha de paquetes OSPF. Además, las actualizaciones sólo se envían en caso de cambios de enrutamiento y no de manera periódica. Esto garantiza un mejor uso del ancho de banda. (6)

1.3.3 Protocolo ICMP

El Protocolo de Mensajes de Control y Error de Internet, ICMP, es de características similares a User Datagram Protocol (UDP), pero con un formato mucho más simple, y su utilidad no está en el transporte de datos de usuario, sino en controlar si un paquete no puede alcanzar su destino, si su vida ha expirado, si el encabezamiento lleva un valor no permitido, si es un paquete de eco o respuesta. Es decir, se usa para manejar mensajes de error y de control necesarios para los sistemas de la red, informando con ellos a la fuente original para que evite o corrija el problema detectado. ICMP proporciona así una comunicación entre el software IP de una máquina y el mismo software en otra. (7)

1.4 Sistemas similares que realizan descubrimiento de la red

1.4.1 AutoScan-Network

AutoScan-Network es un escáner de red. El objetivo principal es imprimir la lista de equipos conectados en red. AutoScan Network es software libre. Este escáner de red está disponible bajo la Licencia Pública General de GNU. Esto significa que es libre de usarlo y modificarlo, pero si distribuye sus modificaciones se debe distribuir el código fuente modificado. (8)

A continuación, se muestran algunas de sus características principales:

- ✓ Rápido escaneo de la red.
- ✓ Descubrimiento automático de la red.
- ✓ Scanner multiproceso.

1.4.2 Open-AudIT

Una herramienta para gestionar el cumplimiento de la política empresarial BYOD (Bring your own device o Trae tu propio dispositivo), el motor de búsqueda de dispositivos escanea rápidamente una red para detectar dispositivos conectados a una red inteligente y almacena las configuraciones de todos los componentes descubiertos. Un marco de información de gran alcance presenta información tal como la concesión de licencias de software, los cambios de configuración, los dispositivos no autorizados, utilización de la capacidad, y el estado de la garantía de hardware. (9)

Las principales características incluyen:

Inventario detallado:

- ✓ Ninguna otra plataforma de auditoría ofrece un inventario tan detallado en cuanto a: agrupación, recopilación, y traducción de los datos en información fácil de consumir.

Seguridad de la base línea de reporte:

- ✓ Permite ver qué dispositivos han cambiado, cuando se produjo el cambio y (cuando sea posible) que hizo el cambio.

Exportación e importación de datos:

- ✓ Los datos de inventario son fácilmente de exportar en una variedad de formatos, incluyendo XML, CSV, JSON, HTML o texto plano. La importación y exportación de datos hacia y desde otros sistemas de gestión es una tarea simple gracias al aprovechamiento de potentes herramientas de importación de datos.

1.4.3 Wireshark

Uno de los sniffer[2] de red y de libre distribución. Se puede ejecutar en Linux, Unix y Windows. Herramienta utilizada para capturar cualquier tráfico que pasa por un interfaz de red. Se deben añadir ciertos componentes a la herramienta para ampliar sus funcionalidades. Entiende todo tipo de protocolo. Si hay un protocolo que no se soporta, la herramienta permite la posibilidad de crear un plugin para entenderlo. Una de sus funcionalidades es el interfaz gráfico que ofrece y la capacidad de implementar filtros por protocolo, ips destino y origen. Permite examinar datos de una red viva o de un archivo de captura salvado en disco. Se puede analizar la información capturada, a través de los detalles y sumarios por cada paquete. Wireshark incluye un completo lenguaje para filtrar lo que se quiere ver y la habilidad de mostrar el flujo reconstruido de una sesión de Transfer Control Protocol (TCP). (10)

1.4.4 Sistema GRHS

Este sistema es una herramienta informática diseñada para mantener el control de los ordenadores conectados a una red de computadoras. Brinda información acerca del hardware y el software que estas poseen, además de sus localizaciones. GRHS,

[2] Sniffer (analizador de protocolos) es un programa de captura de las tramas de una red de computadoras.

cuenta con un cliente instalado en cada PC (Gclient) con el objetivo de realizar el inventario de hardware y software de las computadoras conectadas a la red. Anteriormente se desarrolló un módulo para el descubrimiento de Gclient en las estaciones de trabajo de una subred. Este módulo realiza el escaneo de la red enviando peticiones TCP/IP a cada nodo de la red por un puerto determinado, si logra establecer la conexión por ese puerto, asume que la PC tienen Gclient instalado. Además, escanea las subredes una a una, lo que trae consigo un mal aprovechamiento del tiempo de trabajo y realiza el escaneo de la red basándose en una máscara de subred fija (el sistema usa por defecto la máscara 0/24), característica la cual en algunas circunstancias es necesaria modificar según la necesidad del cliente.

1.5 Conclusiones del estado del arte

Luego de realizar el estado del arte y el estudio del módulo anteriormente desarrollado para el sistema GRHS, se llegó a la conclusión de que las herramientas usadas en el módulo anterior se reutilizarán para la implementación de la nueva versión. Se confirmó el uso de la librería Python-Nmap entre las antes expuestas por reunir las características más generales para el desarrollo del módulo. Python-Nmap es una biblioteca libre que brinda seguridad al servidor, detecta los hosts activos en una subred, además permite escanear los puertos de la PC en cuestión, determinando los servicios que están corriendo por cada uno de ellos. También se selecciona el protocolo ICMP siendo el protocolo interno de la librería antes escogida.

1.6 Metodología de desarrollo:

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Las técnicas indican cómo debe ser realizada una actividad técnica determinada identificada en la metodología. Combina el empleo de unos modelos o representaciones gráficas junto con el empleo de unos procedimientos detallados. (11)

La selección de la metodología a utilizar se hace en base a las características del equipo y las necesidades específicas de la situación. Para elegir una metodología de desarrollo de software se debe tener en cuenta dos factores fundamentales: el tipo de proyecto que se desea desarrollar y el tiempo que se dispone para desarrollar el mismo.

Actualmente existen dos grandes grupos de metodologías de desarrollo, las metodologías tradicionales y las ágiles; las primeras se centran en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto, mientras que las segundas dan mayor importancia a la capacidad de respuesta a los cambios (12). A continuación, se presenta una breve comparación entre ellas.

1.6.1 XP (Programación Extrema)

Es una metodología ágil de desarrollo de software que posee cuatro tareas fundamentales: planificación, diseño, desarrollo y pruebas. Esta metodología está basada en la simplicidad durante el desarrollo, la comunicación entre las partes implicadas (clientes y desarrolladores) y la retroalimentación para poder reutilizar el código desarrollado. En su concepción establece entregas frecuentes con posibilidad de refactorización continua, permitiendo mejorar el diseño cada vez que se añade una funcionalidad. (13)

Se escogió la metodología ágil XP porque el módulo a desarrollar requiere de un corto período de tiempo para su implementación, genera poca documentación, cuenta con un dúo de programadores, el cliente forma parte del equipo de desarrollo y la aplicación se divide en iteraciones donde se le entrega al cliente una versión del módulo.

1.7 Lenguaje de programación

Un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora. (14)

Para la implementación del módulo actual y teniendo en cuenta las tecnologías con las que fue desarrollado GRHS se escoge como lenguaje de programación Python en su versión 2.7.

1.7.1 Python 2.7

Python es un lenguaje de scripting (estructura de guión) independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder

ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

Una ventaja fundamental que presenta dicho lenguaje es la gratuidad de su intérprete. Su intérprete tiene versiones para prácticamente cualquier plataforma en uso: ordenadores con sistema operativo Linux y Windows. (15)

1.8 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. (16) Por las características expuestas a continuación, se elige al SGBD PostgreSQL.

1.8.1 PostgreSQL 9.3

PostgreSQL presenta soporte para gran parte del estándar SQL (Lenguage de Consulta Estructurado), posee características tales como el uso de llaves foráneas, triggers (disparadores), vistas, integridad transaccional, control de concurrencia multiversión y puede ser extendido por el usuario añadiéndole tipos de datos, funciones, operadores, funciones agregadas, métodos de indexado y lenguajes procedurales.

Funciona sobre 34 plataformas incluyendo Windows, Linux, FreeBSD, Solaris, Unix; y además soporta gran cantidad de lenguajes dentro de su API (Application Programming Interface o Interfaz de Programación de Aplicaciones) para el desarrollo de las aplicaciones, como por ejemplo SQL, Java, Perl, Python, C, C++, Ruby y PHP, haciendo mayor énfasis en Python. (17)

1.9 Marco de Desarrollo

Un marco de desarrollo se puede definir como un conjunto de componentes que estructuran un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web como aplicaciones de escritorio. (18) Se seleccionó Django como marco de desarrollo por ser el utilizado en el sistema GRHS.

1.9.1 Django 1.4.5

Django es un marco de desarrollo web de código abierto, escrito en Python, que cumple en cierta medida el paradigma del Modelo Vista Controlador. La meta fundamental de Django es facilitar la creación de sitios web complejos.

Python es usado en todas las partes del marco de desarrollo, incluso en configuraciones, archivos y en los modelos de datos. (19) Se seleccionó el marco de desarrollo Django 1.4.5 para el desarrollo del módulo, por ser utilizado actualmente por el sistema GRHS.

1.10 Entorno de Desarrollo Integrado

Un IDE (Entorno de Desarrollo Integrado o Entorno de Desarrollo Interactivo) es una aplicación de software que proporciona servicios integrales para los programadores informáticos para el desarrollo de software. Un IDE normalmente consiste en un editor de código fuente que permite construir herramientas de automatización. (20) Se decidió utilizar para la reconstrucción del módulo el IDE Pycharm 2016.1 en su versión gratuita:

1.10.1 Pycharm 2016.1

Es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, además brinda soporte para el desarrollo web con Django. (21)

1.11 Herramienta CASE

La herramienta CASE (Computer Aided Software Engineering por sus siglas en inglés e Ingeniería de Software Asistida por Computadora traducido al español) permite Modelar los Procesos de Negocios de las organizaciones. Permite mejorar el diseño de los sistemas, elaborar los requerimientos de los usuarios y garantiza la eficiencia.

1.11.1 Visual Paradigm 5.0

Visual Paradigm es una herramienta CASE que soporta el modelado mediante UML (Unified Modeling Language o Lenguaje Unificado de Modelado) y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Tiene como ventajas que facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto

también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta. (22) Esta herramienta es usada esencialmente para el modelado del proceso de negocio además del modelado de procesos a través de BPMN. Fue necesario el uso de esta herramienta para la representación del modelo de datos y el modelo de proceso.

1.12 Lenguaje de Modelado BPMN

Business Process Modeling Notation (BPMN por sus siglas en inglés) es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades. BPMN proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente. (23) Se escogió esta notación gráfica para describir la lógica de los pasos del modelado del proceso de negocio del módulo a evolucionar.

1.13 UML (Unified Modeling Language) 8.0

El Lenguaje de Modelado Unificado (UML) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s.UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. (24)

1.14 Apache JMeter 2.11

Apache JMeter es una aplicación de código abierto la cual esta desarrollada al 100% en lenguaje Java. Fue diseñada para realizar pruebas de comportamiento a las funciones de un sistema web o simplemente evaluar el rendimiento del mismo (34). A través de esta herramienta fueron diseñadas las pruebas de carga y estrés como parte de los conocidos test de rendimiento incluidos en el plan de pruebas a realizar al módulo después de su desarrollo.

1.15 Conclusiones parciales

Con el desarrollo de este capítulo se logró un entendimiento sobre el módulo en su versión 1.0 así como de la situación problemática planteada en la introducción del

presente trabajo. Con la realización del estudio de las bibliotecas que realizan descubrimiento en la red se determinó el uso de Python-Nmap para el desarrollo del módulo debido a las características que posee, las cuales la hacen la biblioteca propicia para satisfacer las deficiencias del sistema. Se definió XP como la metodología de desarrollo a utilizar. Se decidió utilizar como marco de desarrollo web Django 1.4.5, como lenguaje de programación Python 2.7, PostgreSQL 5.0 como sistema gestor de base de datos, Pycharm 2016.1 como IDE de desarrollo y la notación BPMN para el modelado del proceso de negocio del módulo a implementar.

Capítulo 2: Exploración y Planificación

2.1 Introducción

El presente capítulo está centrado en las fases de Exploración y Planificación de la metodología de desarrollo XP, además de la descripción de la propuesta de solución del presente trabajo de diploma, la cual servirá de gran ayuda durante el desarrollo de las nuevas funcionalidades para el módulo de escaneo de la red.

2.2 Propuesta del sistema

En la búsqueda de una solución a la problemática general de esta investigación se concibe como propuesta, a través de la herramienta Python-Nmap, la reestructuración del módulo existente, de tal forma que sea capaz de favorecer la búsqueda de dispositivos en la red, el cual presenta como características:

1. Posibilidad de configurar los parámetros de escaneo que emplea Python-Nmap con el objetivo de que la búsqueda sea más abarcadora.
2. Simultaneidad de escaneo de subredes para disminuir el tiempo que emplea el servidor en el escaneo.
3. Reconocimiento del servicio Gclient en las PC localizadas en las subredes.
4. Escaneo desde los clientes de GRHS.

Para escanear la red existen dos variantes, la primera de ellas comienza cuando el administrador selecciona las subredes existentes en el sistema. Después de este paso tiene la posibilidad de ejecutar una de las siguientes acciones escanear las subredes escogidas o programar el escaneo de dichas subredes. En la función de programar el escaneo se especificará la fecha a realizar dicha actividad. Al escanear una subred se almacenarán en la base de datos las características correspondientes a cada PC escaneada en su subred, las cuales contienen: dirección IP, si existe el cliente GRHS instalado, sistema operativo y arquitectura del mismo. A continuación, la ilustración 1 describe el proceso explicado anteriormente.

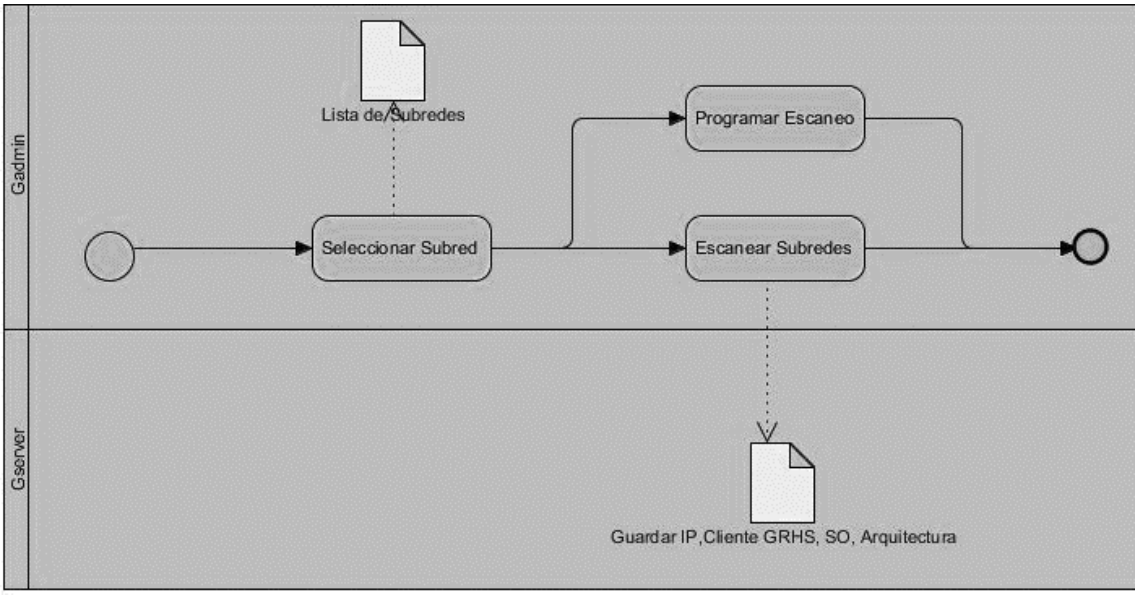


Ilustración 1. Escaneo de la subred por parte del servidor.

La segunda variante para la ejecución del escaneo es iniciada por el Gclient realizando una petición al servidor para comprobar si anteriormente fue escaneada esta subred por otro ordenador, por el servidor o si en ese momento está siendo escaneada. Este intercambio de mensajes se realiza mediante una conexión segura HTTPS. En caso de que la respuesta sea positiva el proceso termina, sino Gclient continúa su ejecución encuestando la subred a la cual pertenece y enviando los datos al servidor. La imagen que se muestra a continuación muestra la modelación del proceso descrito.

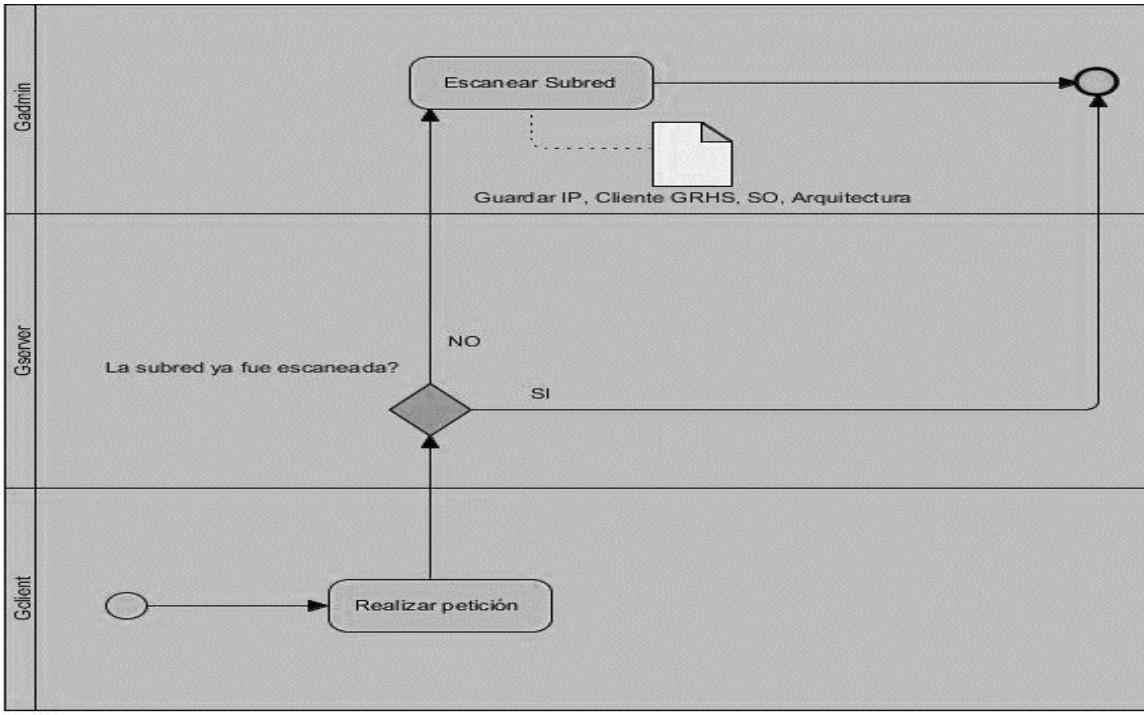


Ilustración 2. Escaneo de la subred por parte del Gclient

2.3 Funcionalidades

Las nuevas funcionalidades que se desarrollarán serán las siguientes:
F1. Seleccionar subredes a escanear.
F2. Configurar parámetros para el escaneo.
F3. Realizar escaneo simultáneo de las subredes.
F4. Chequear si fue instalado el cliente de GRHS en cada ordenador.
F5. Permitir el escaneo de la subred empleando Gclient.
F6. Escaneo programado y automático de subredes específicas.

2.4 Propiedades del producto

Las propiedades del producto son las cualidades que todo sistema debe poseer para un correcto funcionamiento. A continuación, se definen los siguientes requisitos:

Usuario Final

- ✓ El usuario final debe tener conocimiento del proceso de negocio según el papel que ocupa en el sistema y poseer conocimiento básico del uso de la computadora.

Diseño e Implementación

- ✓ El módulo debe ser implementado utilizando el lenguaje de programación Python 2.7, el marco de trabajo Django 1.4.5 y la biblioteca Python-Nmap para su desarrollo.

Software

- ✓ Los ordenadores donde serán ejecutados los clientes de GRHS debe tener instalado Mozilla Firefox 3.X o superior. El servidor para el manejo de la base de datos debe contar con PostgreSQL 9.3 o una versión superior.

Requisitos de Hardware

- ✓ La PC donde estará instalado el sevidor GRHS debe contar con un microprocesador de 3 GHz, memoria RAM 2 GB y espacio de almacenamiento 500 GB.

Requisitos de Seguridad

- ✓ Siguiendo los estándares de seguridad establecidos en el sistema GRHS, el módulo fue desarrollado en el marco de trabajo Django, basándose en la arquitectura Modelo Vista Plantilla (MVP), para de este modo garantizar el acceso únicamente a las personas autorizadas.

2.5 Historias de usuario

La metodología XP está basada fundamentalmente en las Historias de Usuario (HU), las cuales representan los requisitos del sistema y a través de estas el cliente describe sus necesidades. Las HU son descripciones breves sin palabras rebuscadas de manera tal que se obtenga una idea básica de qué tiempo durará el desarrollo del sistema. También proporcionan detalles sobre la estimación de riesgos. (25)

Las HU se componen de un número de HU, nombre de la historia de usuario, ocurrencia de modificación, usuario que realiza la acción, la iteración asignada, la prioridad para el negocio, los puntos estimados, los puntos reales, el nivel de riesgo existente en el desarrollo, programador(es) encargado(s) de la implementación, una descripción, observaciones y un prototipo de interfaz en caso de que aplique.

A continuación, se exponen las HU definidas por el cliente en conjunto con equipo de desarrollo:

<table>
<tr><td colspan="3">Historia de Usuario</td></tr>
<tr><td>Número: 1</td><td colspan="2">Nombre Historia de Usuario: Seleccionar subredes a escanear.</td></tr>
<tr><td colspan="2">Usuario: Administrador</td><td>Iteración Asignada:1</td></tr>
<tr><td colspan="2">Prioridad en Negocio: Alta</td><td>Puntos Estimados: 1</td></tr>
<tr><td colspan="3">Riesgo en Desarrollo: Alto</td></tr>
<tr><td colspan="3">Programador(es) responsable(s): Daniel Martínez Parra

Rafael González Arrieta</td></tr>
</table>

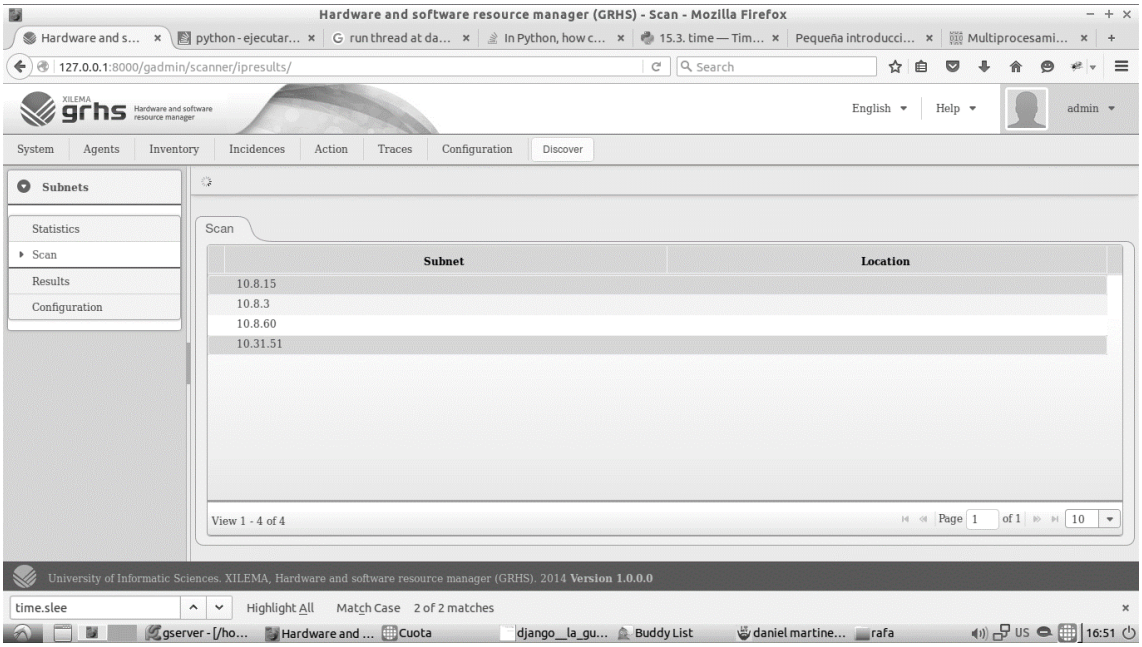
Descripción: A partir de las subredes almacenadas en la base de datos se seleccionan desde Gadmin las subredes a escanear.
Observaciones
Prototipo de Interfaz: 

Tabla 1. HU1: Seleccionar subredes a escanear.

Historia de Usuario	
Número: 2	**Nombre de Historia de Usuario:** Configurar parámetros para el escaneo.
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Alto	

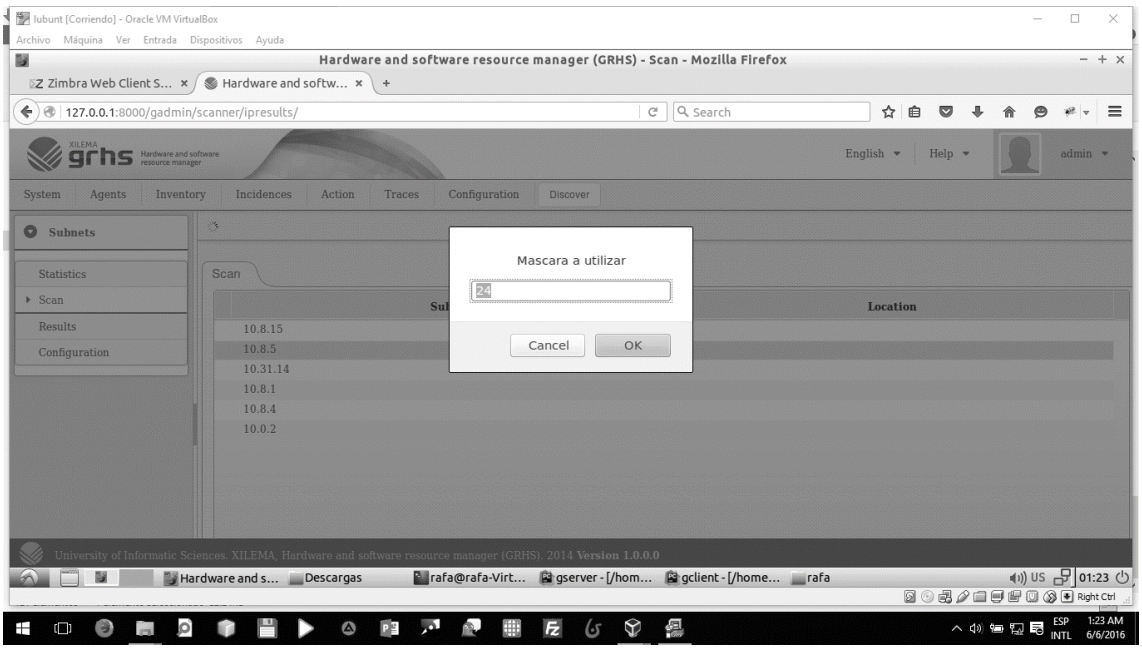
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta
Descripción: Modificar los parámetros de la librería Nmap, correspondientes a la máscara de red que se desea escanear.
Observaciones
Prototipo de Interfaz: 

Tabla 2. HU2: Configurar parámetros que utiliza Python-Nmap para el escaneo.

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Realizar escaneo simultáneo de las subredes.
Usuario: Administrador	**Iteración Asignada:** 2

Prioridad en Negocio: Media	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	
Descripción: Realizar el escaneo simultáneo de todas las subredes almacenadas en la lista de escaneo.	
Observaciones: El administrador tiene la opción de escoger una o varias subredes a escanear.	
Prototipo de Interfaz: NA	

Tabla 3. HU3: Realizar escaneo simultaneo de las subredes.

Historia de Usuario	
Número: 4	**Nombre Historia de Usuario:** Chequear si fue instalado el cliente GRHS en cada ordenador.
Usuario: Administrador	**Iteración Asignada:** 2
Prioridad en Negocio: Media	**Puntos Estimados:** 2
Riesgo en Desarrollo: Alto	
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	

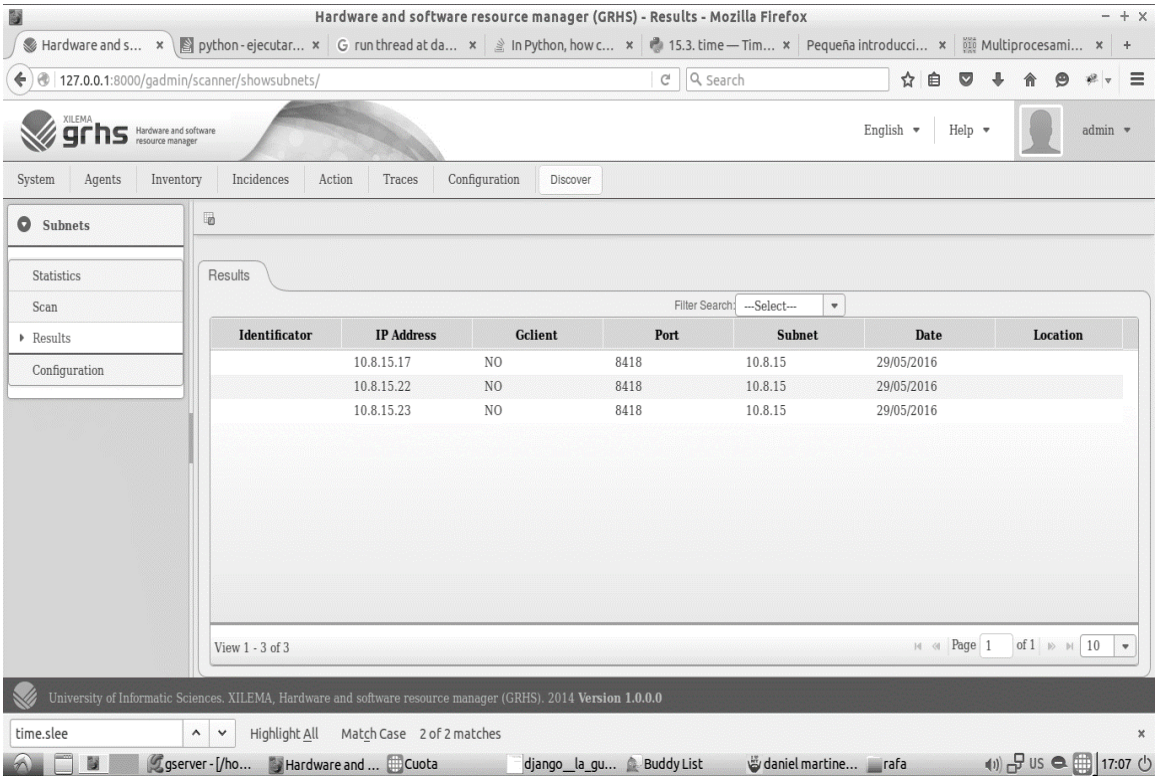
Descripción: Se escanea la subred determinando por cada IP si existió algún antecedente de conexión entre el Gclient ubicado en esa PC y el Gserver.
Observaciones: Existe la posibilidad que el puerto 8418 este abierto por otra aplicación y no necesariamente esté instalado el cliente de GRHS.
Prototipo de Interfaz: 

Tabla 4. HU4: Chequear si fue instalado el cliente GRHS en cada ordenador.

Historia de Usuario	
Número: 5	**Nombre Historia de Usuario:** Permitir el escaneo de la subred del lado del Gclient.
Usuario: Administrador	**Iteración Asignada:** 2
Prioridad en Negocio: Baja	**Puntos Estimados:** 1

Riesgo en Desarrollo: Alto
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta
Descripción: El cliente escanea su subred si esta no ha sido escaneada anteriormente.
Observaciones:
Prototipo de Interfaz: NA

Tabla 5. HU5: Permitir el escaneo de la subred del lado del cliente.

Historia de Usuario	
Número: 6	**Nombre Historia de Usuario:** Permitir escaneo programado y automático de subredes específicas.
Usuario: Administrador	**Iteración Asignada:** 1
Prioridad en Negocio: Baja	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	
Descripción: Se realiza el escaneo de las subredes seleccionadas de forma automática una vez que ha sido programado por el administrador.	
Observaciones:	

Prototipo de

Interfaz:

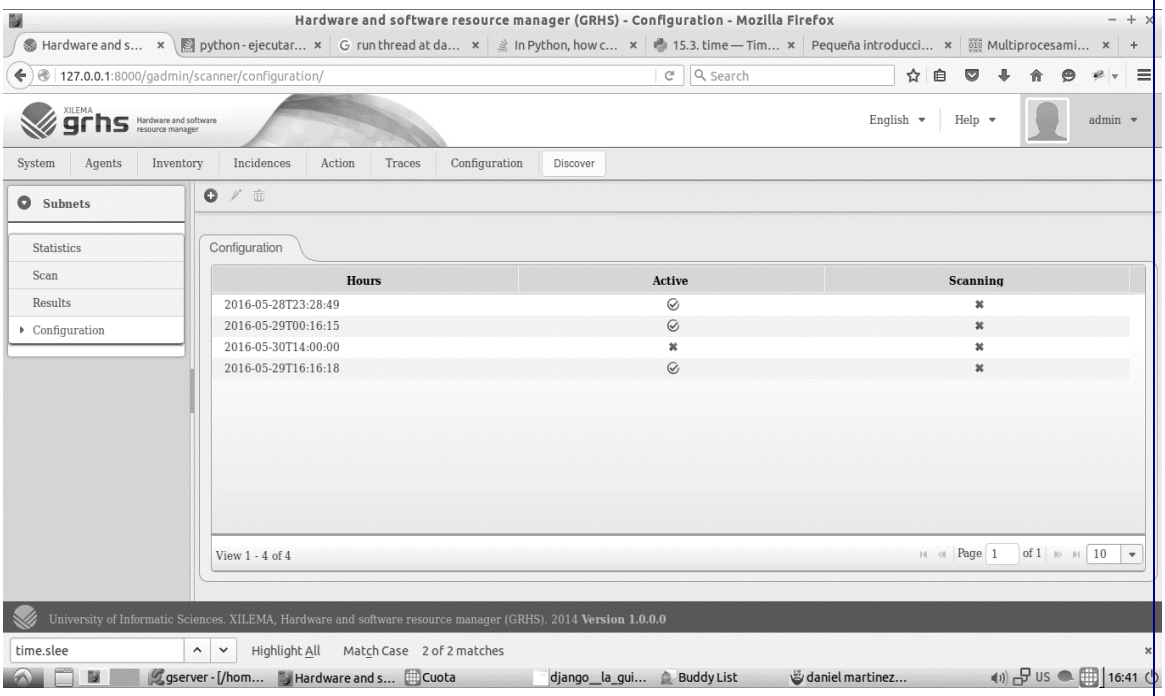


Tabla 6. HU6: Permitir el escaneo de la subred del lado del cliente.

2.6 Estimación de esfuerzo de trabajo

La estimación de esfuerzo de trabajo se basa en expresar por medio de puntos, momentos donde el trabajo que se planeaba realizar se concluyeran sin ninguna interrupción, o sea, un punto es considerado como un día de trabajo ideal. Dicha estimación también incluye cualquier esfuerzo asociado al desarrollo ubicado en la historia de usuario. (25)

Historias de usuario	Puntos de estimación
Seleccionar subredes a escanear.	1
Configurar parámetros para el escaneo.	2
Realizar escaneo simultáneo de las subredes.	1
Chequear si fue instalado el cliente GRHS en cada ordenador.	2
Permitir el escaneo de la subred del lado del cliente.	1

Escaneo programado y automático de subredes específicas.	1

Tabla 7. Estimación de esfuerzo por HU

2.7 Plan de iteraciones

A partir de las descripciones de las historias de usuarios y la estimación de esfuerzo correspondiente se procede a realizar el plan de iteraciones:

Iteración 1: Esta iteración tiene como objetivo la implementación de las historias de usuario dedicadas a la pre-configuración de escaneo.

Iteración 2: Esta iteración se basa en la implementación de la selección de subredes, programación de eventos y el escaneo por parte del cliente.

Iteración 3: En esta iteración se implementarán las historias de usuario relacionadas con el escaneo por parte del servidor.

2.8 Plan de duración de iteraciones

Iteraciones	**Orden de historias de usuario a desarrollar**	**Duración total de la iteración**
Iteración 1	- Configurar parámetros para el escaneo.	2
Iteración 2	- Seleccionar subredes a escanear. - Escaneo programado y automático de subredes específicas. - Permitir el escaneo de la subred del lado del cliente.	3
Iteración 3	- Realizar escaneo simultáneo de las subredes. - Chequear si fue instalado el cliente GRHS en cada ordenador.	3

Tabla 8. Plan de Iteraciones

2.9 Plan de entregas

El plan de entregas no es más que una definición de cada una de las entregas de la solución que se le entregará al cliente con su respectiva fecha, acordado previamente con este, en reuniones iniciales del inicio del proyecto. **(25)**

Iteración 1	**Iteración 2**	**Iteración 3**
25% del módulo 28/04/2016	70% del módulo 17/05/2016	100% del módulo 1/06/2016

Tabla 9. Plan de entregas

2.10 Conclusiones parciales

Con el desarrollo de este capítulo se logra una mejor visión del producto final y el contenido que abarca en cuanto a escaneo de subredes, representado de forma visual el modelo de proceso para un mayor entendimiento. Se identificaron las funcionalidades y las propiedades del producto. Se especificó el sistema de planificación teniendo en cuenta las historias de usuario y las iteraciones. Una vez culminado ya se puede pasar al diseño e implementación del módulo.

Capítulo 3: Diseño, Implementación y Pruebas

3.1 Introducción

En el transcurso del presente capítulo se describe el proceso de implementación de la propuesta que da solución al problema planteado, mostrando así características que componen el desarrollo del diseño del módulo. Estas características incluyen patrones de arquitectura y diseño, según la metodología de desarrollo XP, además de las tarjetas Clase – Responsabilidad – Colaborador (CRC) que contribuyen a las tareas del diseño. Las tareas de ingeniería constituyen pasos indispensables a seguir para desarrollar el módulo.

3.2 Arquitectura de software

La arquitectura del software indica la estructura, funcionamiento e interacción entre las partes del software. Es un nivel de diseño que hace énfasis en aspectos más allá de la implementación del código en sí, sino como los componentes de una aplicación se comunican entre ellas a través de las interfaces que utilizan. (25)

3.2.1 Arquitectura Cliente-Servidor

Esta arquitectura está basada en un diseño de aplicación distribuida en la que las tareas realizadas en todo el tiempo de ejecución del software, son repartidas entre los servidores, sistemas encargados de brindar recursos, servicios, y los clientes, llamados a consumir. Presenta un funcionamiento básico, el cliente realiza una petición y el servidor le da respuesta. (26)

En el proyecto GRHS esta arquitectura se utiliza de tal manera que el servidor puede estar en más de una computadora y la aplicación puede estar corriendo en varias instancias. El módulo funciona a través de los servicios Gserver y Gclient, una vez instalados en el servidor y el cliente en ese mismo orden.

3.2.2 Patrón Modelo-Vista-Plantilla

En el desarrollo del sistema se utilizó el patrón llamado MVT, que en Django hace referencia a modelo vista plantilla, el modelo sigue siendo el modelo, en este caso, la vista no es una vista, sino que más bien es un controlador que se llama vista y la plantilla son las vistas del MVP, es decir, los formularios van en la plantilla, los

formularios hacen peticiones a las vistas y las vistas obtienen datos de los modelos. (27)

Básicamente los 3 componentes principales del marco de desarrollo tienen como función crear una comunicación entre ellos para extraer información vital de la base de datos y presentarlas en el navegador.

Modelo: El modelo tiene como objetivo mapear la base de datos de tal forma que cree una sincronización entre la base de datos y la aplicación, con el fin de mantener actualizada toda la información de las tablas, campos y datos de nuestra base.

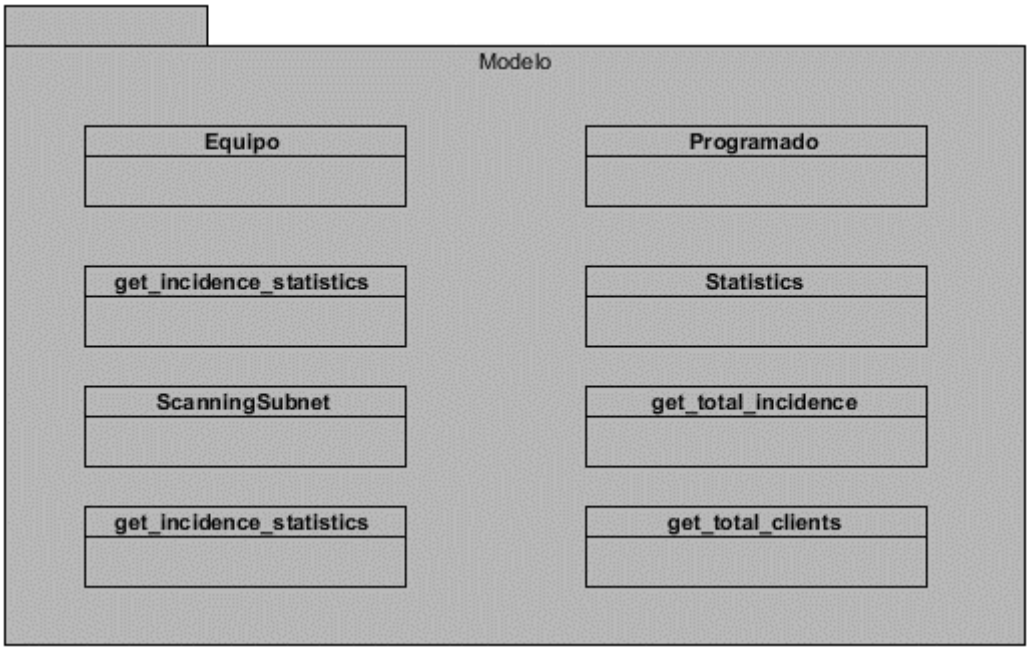


Ilustración 3. Capa Modelo

Vista: La vista tiene como objetivo recibir el requerimiento que es enviado por el navegador, procesar la información, si es necesario realizar una petición al modelo para extraer un valor de base de datos u otro de los casos podría ser que sólo muestre un mensaje enviando un requerimiento a la plantilla.

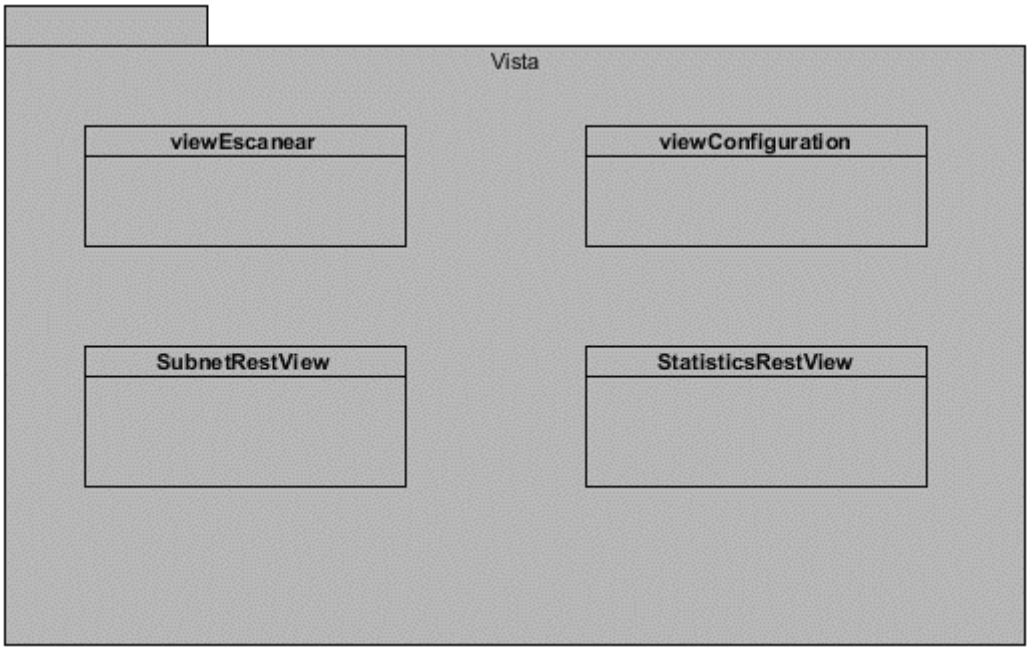


Ilustración 4. Capa Vista

Plantilla: Es básicamente un archivo html que tiene como objetivo mostrar en pantalla de forma amigable la respuesta que es enviada por la vista.

3.3 Patrones de Diseño

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.

Patrones GRASP

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Patrón Experto: se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos. Hace referencia a las clases que contienen la información necesaria para cumplir con las responsabilidades que le fueron asignadas. Se evidencia el uso de este patrón en la clase Equipo, la misma contiene los datos los cuales son necesarios para cumplir con su responsabilidad, en este caso, almacenar los datos de las computadoras conectadas en cada subred, para posteriormente ser analizada esta información por el administrador.

A continuación, el ejemplo descrito anteriormente, tomado del archivo models.py:

```
class Equipo(models.Model):
    mb = models.CharField(null=False, max_length=50)
    ip = models.CharField(null=False, max_length=50)
    grh = models.CharField(null=False, max_length=50)
    subnet = models.CharField(null=False, max_length=50)
    dat = models.CharField(null=False, max_length=50)
    location = models.CharField(null=False, max_length=50)
```

Ilustración 5. Ejemplo del uso del Patrón Experto en la implementación

Patrón Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica de este patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Es el encargado de crear las instancias de los objetos para cada una de las funcionalidades de la aplicación. Se toma como ejemplo desde el metodo **escanear**, este fragmento del código responsable de crear un objeto de la clase Equipo para que sea guardada en la base de datos y luego mostrarla al usuario.

Ejemplo de utilización del patrón:

```
                disc = Equipo(mb=str(iden),
                              ip=str(host),
                              grh=str(gclient),
                              subnet=str(self.subnet),
                              dat=str(dat),
                              location=str(location))
                disc.save()
```

Ilustración 6. Ejemplo del uso del Patrón Creador en la implementación

Patrón Alta cohesión: se refiere al grado o la fuerza con que se relacionan algunos elementos, es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Un elemento con alta cohesión, realiza tareas relacionadas entre sí. Se evidencia en el uso de las clases BusquedaSubred y BusquedaCompleta, donde se les asignan responsabilidades para que logren realizar el trabajo en conjunto.

Patrón Bajo Acoplamiento: cuando se habla de acoplamiento entre objetos, se hace referencia a la "fuerza" con la que ciertos objetos están relacionados, o dependen unos de otros. Mientras más dependencias tengan un objeto de otros para llevar a cabo sus tareas, más fuerte será el acoplamiento. Se le asigna esta responsabilidad a la clase StatisticSerializer, responsable de crear una nueva instancia de las clases Statistic para la visualización por el usuario.

3.4 Tarjetas CRC

La utilización de tarjetas CRC (Clase Responsabilidad Colaborador) es una técnica de diseño orientado a objetos. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que vamos a necesitar para implementar el sistema y la forma en que van a interactuar. (25)

A continuación se representan las tarjetas CRC correspondientes al módulo:

Clase: BusquedaSubredes	
Responsabilidad	**Colaborador**
Devolver de la subred las PC encendidas y sus características principales, para el posterior escaneo.	NetworkInterface Escanear

Tabla 10. Tarjeta CRC: Subredes

Clase: BusquedaProgramada	
Responsabilidad	**Colaborador**
Proporciona las subredes para el escaneo automático y programado de las subredes seleccionadas	Escanear

Tabla 11. Tarjeta CRC: Escaneo

Clase: NetworkInterface	
Responsabilidad	**Colaborador**

Almacena las subredes en la base de datos	Save

Tabla 12. Tarjeta CRC: NetworkInterface

Clase: viewEscanear	
Responsabilidad	**Colaborador**
Suministra la información de las subredes una vez escaneadas para luego ser mostrados al usuario los resultados.	SubnetRestView StatisticsRestView viewShowSubnets

Tabla 13. Tarjeta CRC: viewEscanear

3.5 Modelo de la Base de Datos

Un modelo de datos es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de reglas de integridad, las cuales definen un conjunto de estados consistentes. (28) La imagen presentada a continuación hace referencia al modelo físico de la base de datos del módulo reconstruido para el sistema

GRHS:

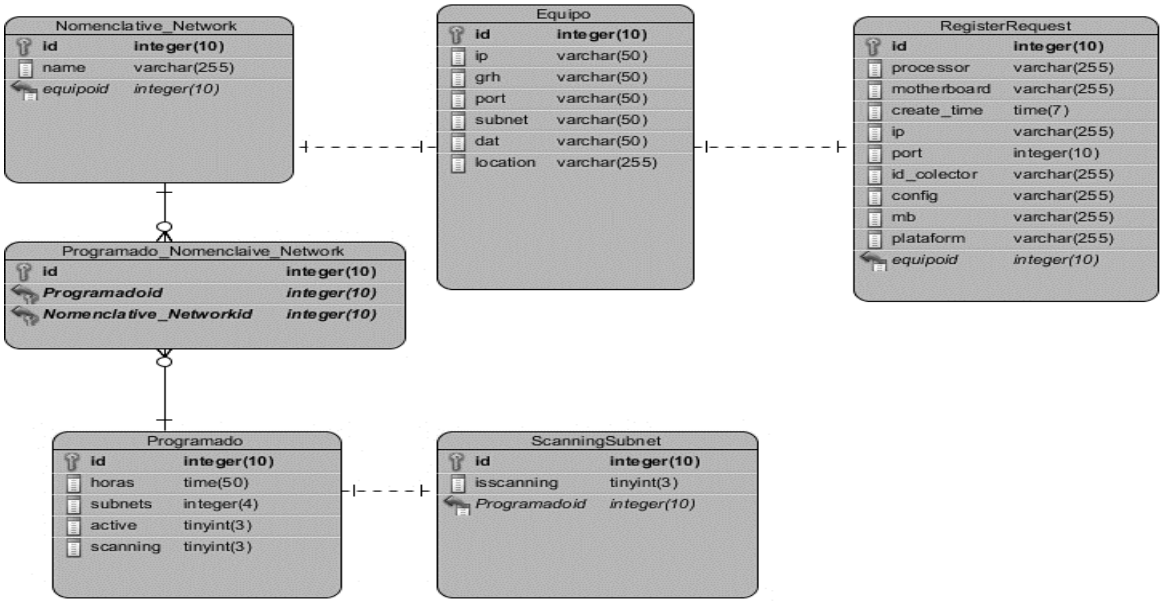


Ilustración 7. Modelo Físico de la Base de Datos

Equipo: Muestra los resultados obtenidos una vez realizado el escaneo de las subredes.

Programado: Esta entidad recolecta la configuración de las subredes programadas para ser escaneadas.

ScanningSubnet: Almacena la información relacionada con el proceso del escaneo automático, si se está escaneando una subred o no.

3.6 Tareas de Ingeniería

Las Tareas de Ingeniería (TI) se realizan muy de la mano de las Historias de Usuario. Estas describen de manera más exacta los pasos que deben realizarse para organizar el trabajo de desarrollo de cada uno de los requisitos funcionales plasmados en las HU. Definen además la fecha, según el tiempo estimado, en que debe comenzar y concluir la realización de la tarea. El correcto tratamiento de este artefacto brinda al equipo de desarrollo una visión estructurada de lo que es necesario para obtener el producto en el tiempo especificado y con la calidad mínima deseada. (25)

A continuación se muestran las TI correspondientes a cada HU:

Tarea de Ingeniería	
Número de Tarea: 1	**Número de Historia de Usuario:**1
Nombre de la Tarea: Seleccionar subredes.	
Tipo de Tarea: Desarrollo (Desarrollo/Corrección/Mejora)	**Puntos Estimados:**1
Fecha Inicio: 5/03/2016	**Fecha Fin:** 12/03/2016
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	
Descripción: Desde Gadmin, el usuario tiene la posibilidad de seleccionar las subredes destinadas a ser escaneadas por el sistema. Las subredes a escanear cubren valores desde una hasta el número total de subredes listadas.	

Tabla 14. Tarea de Ingeniería 1: Seleccionar subredes.

Tarea de Ingeniería	
Número de Tarea: 2	**Número de Historia de Usuario:**2
Nombre de la Tarea: Configurar parámetros para el escaneo.	
Tipo de Tarea: Desarrollo (Desarrollo/Corrección/Mejora)	**Puntos Estimados:**2
Fecha Inicio: 13/03/2016	**Fecha Fin:** 27/03/2016

Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta
Descripción: El usuario puede cambiar el valor de la máscara a utilizar a través de Gadmin de forma manual, permitiendo escanear cualquier máscara de subred a la que esté conectado el sistema. Esto se logra gracias al uso de la librería IPcalc, la cual proporciona a la librería Nmap un listado con todas las direcciones IP contenidas en una subred a escanear según el valor de la máscara suministrado.

Tabla 15. Tarea de Ingeniería 2: Configurar parámetros de Python-Nmap.

Tarea de Ingeniería	
Número de Tarea: 3	**Número de Historia de Usuario:**3
Nombre de la Tarea: Escaneo simultaneo de las subredes.	
Tipo de Tarea: Desarrollo (Desarrollo/Corrección/Mejora)	**Puntos Estimados:**1
Fecha Inicio: 28/03/2016	**Fecha Fin:** 4/04/2016
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	
Descripción: El servidor realiza el escaneo de todas las subredes almacenadas en la lista a escanear, actualizando en tiempo real la condición de cada PC si esta se encuentra encendida.	

Tabla 16. Tarea de Ingeniería 3: Escaneo simultáneo de las subredes.

Tarea de Ingeniería	
Número de Tarea: 4	**Número de Historia de Usuario:**4
Nombre de la Tarea: Verificar que haya sido instalado el cliente GRHS.	
Tipo de Tarea: Desarrollo (Desarrollo/Corrección/Mejora)	**Puntos Estimados:**2
Fecha Inicio: 05/04/2016	**Fecha Fin:** 19/04/2016
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	
Descripción: Se realiza el escaneo de la subred accediendo a cada PC y comprobando que la dirección IP del PC ha sido registrada anteriormente en el sistema a través del Gclient, si no es el caso, el módulo entiende que el servicio Gclient no está corriendo correctamente en dicha dirección, en caso contrario, asume que el servicio está funcionando de forma satisfactoria.	

Tabla 17. Tarea de Ingeniería 4: Verificar que haya sido instalado el cliente GRHS.

Tarea de Ingeniería	
Número de Tarea: 5	**Número de Historia de Usuario:**5
Nombre de la Tarea: Escanear la subred por parte del cliente.	
Tipo de Tarea: Desarrollo (Desarrollo/Corrección/Mejora)	**Puntos Estimados:**1

Fecha Inicio: 20/04/2016	**Fecha Fin:** 27/04/2016
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	
Descripción: El cliente comprueba con el servidor si no hay actualmente otra instancia de escaneo en su subred y si no ha sido escaneada anteriormente, en caso contrario, comienza el escaneo de la misma.	

Tabla 18. Tarea de Ingeniería 5: Escanear la subred por parte del cliente.

Tarea de Ingeniería	
Número de Tarea: 6	**Número de Historia de Usuario:**6
Nombre de la Tarea: Realizar el escaneo programado y automático de las subredes específicas.	
Tipo de Tarea: Desarrollo (Desarrollo/Corrección/Mejora)	**Puntos Estimados:**1
Fecha Inicio: 28/04/2016	**Fecha Fin:** 05/05/2016
Programador(es) responsable(s): Daniel Martínez Parra Rafael González Arrieta	
Descripción: Se especifica en el sistema la fecha en la que el administrador desea realizar el escaneo de las subredes seleccionadas, estas tareas se realizarán de forma automática por parte del sistema una vez que haya sido programado.	

Tabla 19. Tarea de Ingeniería 6: Realizar el escaneo programado y automático de las subredes específicas.

3.7 Pruebas de Software

Una vez generado el código fuente, es necesario probar el software para descubrir y corregir la mayor cantidad de errores posibles antes de entregarlo al cliente. El objetivo de las pruebas es procesar la ejecución de un programa con el fin de descubrir un error, con poca cantidad de tiempo y esfuerzo, con un buen caso de prueba se amplía más la alta probabilidad de encontrar un error, con el éxito de una prueba se descubren errores no detectados con anterioridad.

Las pruebas de software es artefacto que poseen los ingenieros de software para determinar el status real de la calidad de un producto. En este proceso se ejecutan pruebas dirigidas a componentes del sistema o al sistema en su totalidad midiendo con esto el grado en que el software cumple con los requerimientos.

3.7.1 Niveles de pruebas

Existen varios tipos de pruebas a ser realizados, estas pueden ser: las de aceptación, las unitarias y las de rendimiento. Las pruebas de aceptación fueron realizas para validar el desarrollo del componente de visualización de selección por parte del administrador de las subredes a escanear. Las unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcionen correctamente por separado. Consta en realizar pruebas para verificar que gran conjunto de partes de software funcionan juntos. Las pruebas de rendimiento se dividieron en tres secciones, Prueba de carga, de estrés y de desempeño para valorar la calidad y eficiencia del software. (29)

3.7.2 Pruebas de aceptación

El uso de cualquier producto de software debe estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada 'prueba de aceptación'. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Especifican, desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente, y que la aplicación cumpla lo realmente solicitado por el cliente. La ejecución de las pruebas diseñadas permitió la evaluación del módulo en desarrollado antes de implantarlos en su entorno de explotación.

Las pruebas de aceptación final del cliente son las encargadas de asegurar el cumplimiento de los procesos elementales del negocio, los casos de uso, lo aceptado en los prototipos y lo planteado en el proyecto técnico.

A continuación se muestra un ejemplo de una de las prueba de aceptación:

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
El administrador del sistema selecciona una o varias subredes.		El sistema muestra las subredes seleccionadas por el administrador.	Satisfactorio.	
Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
	El administrador no selecciona ninguna subred.	El sistema no realizara ninguna acción, en otro caso muestra una alerta informando que debe seleccionar al menos una subred	Satisfactorio.	

Tabla 20. Prueba de aceptación de la HU: Seleccionar subredes a escanear.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
El administrador del sistema inserta la máscara de red que desea escanear.		Se guarda la máscara de red insertada en la configuracion del módulo.	Satisfactorio.	

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la	Observaciones
	El administrador del sistema no inserta ningun elemento o inserta elementos invalidos, como letras o caracteres especiales.	El sistema muestra una alerta informando que debe insertar correctamente los datos.	Satisfactorio.	

Tabla 21. Prueba de aceptación de la HU: Configurar parámetros para el escaneo.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
Las subredes que se desean escanear son seleccionadas por el administrador.		El sistema indica el escaneo en progreso .	Satisfactorio.	
Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
	El administrador no selecciona ninguna subred a escanear.	El sistema muestra una alerta informando que debe seleccionar al menos una subred a escanear.	Satisfactorio.	

Tabla 22. Prueba de aceptación de la HU: Realizar escaneo simultáneo de las subredes.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones

El sistema escanea las subredes accediendo a cada una de las PC que se encuentran encendidas.		El sistema muestra en tiempo real los resultados del escaneo.	Satisfactorio.	
Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
	Si existe alguna PC apagada en la subred no se muestran los resultados.	De las PC que estan apagadas en la subred no se muestra ningún resultado.	Satisfactorio.	

Tabla 23. Prueba de aceptación de la HU: Chequear si fue instalado el cliente GRHS en cada ordenador.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Gclient inicia el escaneo de su subred de forma automática.		Gclient manda los resultados al servidor mostrándose en Gadmin.	Satisfactorio.	
Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
	Si la subred fue escaneada anteriormente Gclient no realiza ninguna acción.	No se muestra ningún resultado en Gadmin.	Satisfactorio.	

Tabla 24. Prueba de aceptación de la HU: Permitir el escaneo de la subred del lado del Gclient.

Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**

El administrador del sistema inserta la fecha en que desea escanear las subredes seleccionadas.		A partir de la fecha especificada por el administrador, el sistema inicia el escaneo de las subredes.	Satisfactorio.	
Clases Válidas	**Clases Inválidas**	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
	El administrador del sistema inserta letras o caracteres especiales.	El sistema muestra una alerta indicando entrar correctamente los datos.	Satisfactorio.	

Tabla 25. Prueba de aceptación de la HU: Permitir escaneo programado y automático de subredes específicas.

3.7.3 Pruebas de Caja Blanca

Las pruebas de caja blanca también se conocen como pruebas de caja de cristal o pruebas estructurales y son una técnica de verificación que cada ingeniero de software puede usar para conocer si el procedimiento seguido por un código es correcto (30).

Se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. Esta prueba verifica la correcta implementación de las unidades internas, las estructuras y sus relaciones y hacen énfasis en la reducción de errores internos (30).

Los métodos de caja blanca o estructural permiten derivar casos de prueba que garanticen que todas las rutas independientes dentro del módulo se ejecuten al menos una vez, que se ejecuten los lados verdadero y falso de todas las decisiones lógicas, ejecuten todos los ciclos dentro y en sus límites operacionales y se ejerciten las estructuras de datos internas para asegurar su validez.

Entre los tipos de prueba de caja blanca que existen se encuentran:

- ✓ **Prueba de condición:** Método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo del programa.

- ✓ **Prueba de flujo de datos:** Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- ✓ **Prueba del camino básico:** Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.
- ✓ **Prueba de bucles:** Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

En este caso se aplicará la llamada **prueba de camino básico** con el objetivo de comprobar que cada línea de código se ejecuta al menos una vez, es decir, obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

La prueba del camino básico es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (25).

A continuación, se definen los pasos necesarios para realizar dicha prueba:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de pruebas que obliguen a la ejecución de cada camino del conjunto básico.

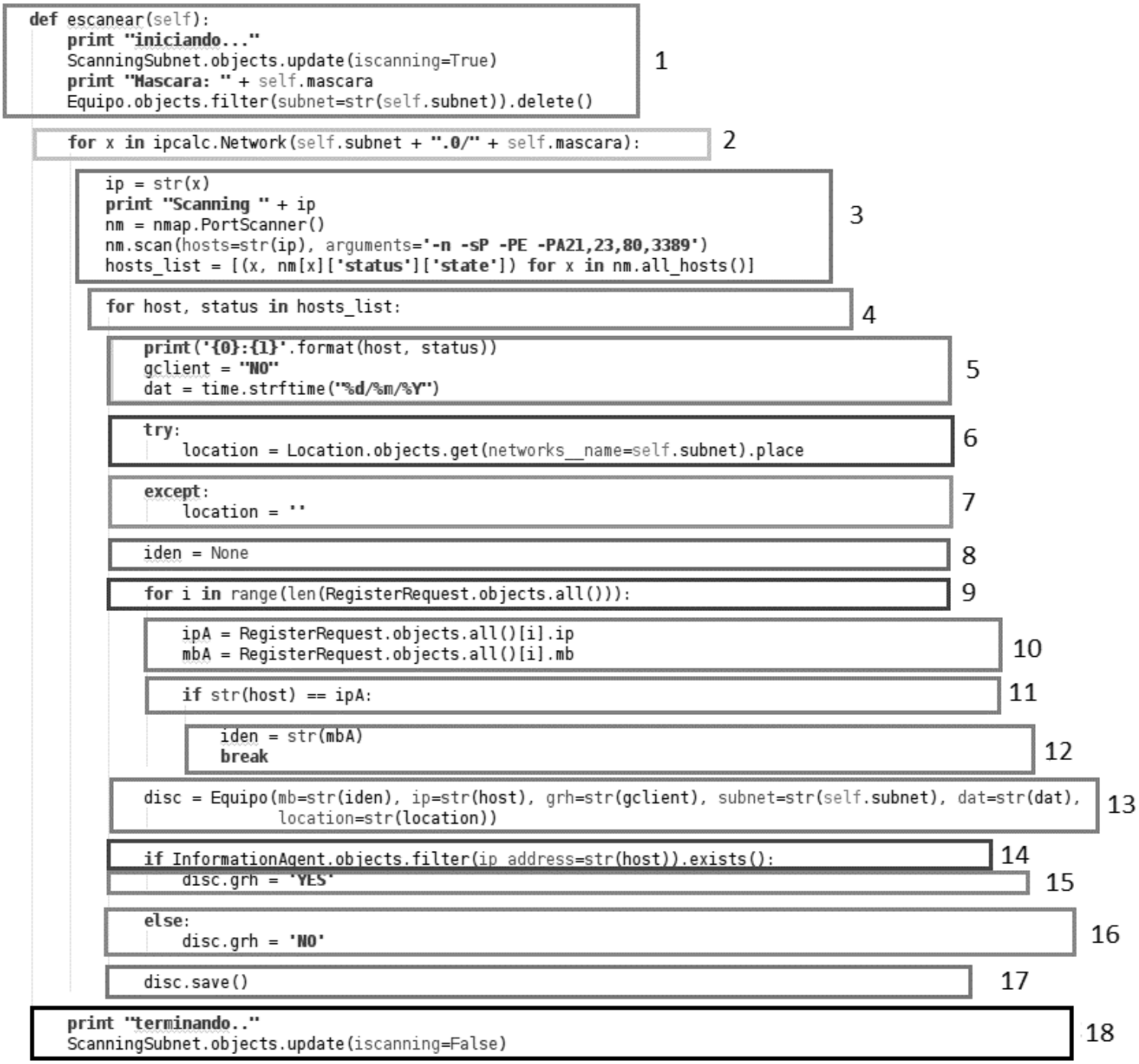
El grafo de flujo se constituye por tres componentes fundamentales:

Nodo: Es usado para representar una o más secuencias procedimentales, es decir, representa cada línea de código que pertenece a la funcionalidad a evaluar.

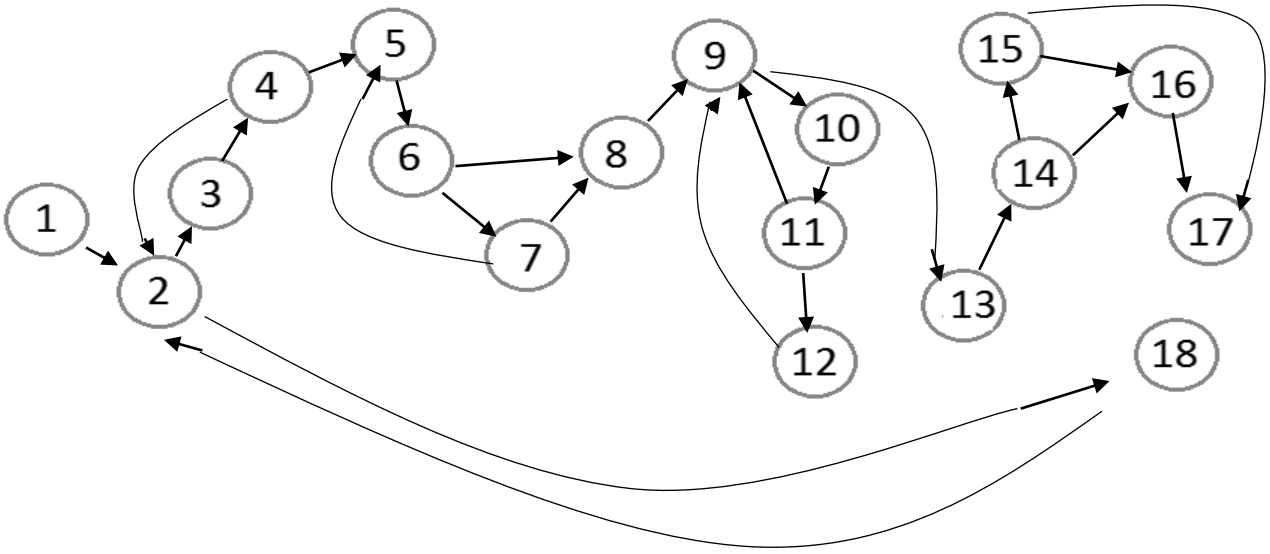
Arista: Es la relación entre cada nodo, implica el sentido del flujo de la ecuación de cada línea de código.

Regiones: Son áreas cerradas delimitadas por aristas, estas representan la cantidad de caminos independientes al ejecutarse un programa.

Para realizar la prueba de camino básico es necesario seleccionar una funcionalidad del sistema, en este caso se seleccionó la funcionalidad **escanear**, por ser considerada una funcionalidad importante para el funcionamiento de la herramienta. A continuación, se muestra el código y selección de las áreas que representan cada nodo:



Continuando con la aplicación de la prueba se construye un grafo el cual representa el flujo de la funcionalidad y cada arista indica todos los posibles caminos a seguir.



La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se utilizan en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática, define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (25). A partir del grafo anterior se obtiene la complejidad ciclomática, esta cuenta con tres fórmulas las cuales deben dar el mismo resultado, las fórmulas son:

1. V (G) = (A – N) + 2, donde “A” es el número de aristas y “N” el de nodos.
2. V (G) = P + 1, donde “P” es la cantidad de nodos predicados (son los nodos de los cuales parten 2 o más aristas).
3. V (G) = R, donde “R” es la cantidad total de regiones contando la región exterior.

Obteniendo como resultado:

1. V (G) = (25 – 18) + 2 = 9
2. V (G) = 8 + 1 = 9
3. V (G) = 9

Los cálculos anteriormente efectuados dan como resultado que la complejidad ciclomática es de 9, a partir de aquí se conoce que existen 9 posibles caminos por donde pude circular el flujo y a su vez a conocer el número de casos de prueba a realizar para asegurar que cada sentencia se ejecute al menos una vez. Los caminos básicos a recorrer son:

CB1. 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 9, 13, 14, 15, 17, 2, 18

CB2. 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 9, 13, 14, 16, 17, 2, 18

CB3. 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 9, 13, 14, 15, 17, 2, 18

CB4. 1, 2, 3, 4, 5, 6, 8, 9, 13, 14, 15, 17, 2, 18

CB5. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 9, 13, 14, 15, 17, 2, 18

CB6. 1, 2, 3, 4, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 9, 13, 14, 15, 17, 2, 18

CB7. 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 9, 13, 14, 16, 17, 2, 18

CB8. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 9, 13, 14, 15, 17, 2, 18

CB9. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 9, 13, 14, 16, 17, 2, 18

Luego de obtener los caminos básicos es necesario realizar los casos de prueba, debe obtenerse al menos uno por cada camino definido, a continuación se muestra un ejemplo de estos:

La realización de los casos de prueba se ejemplificará con la descripción de un caso de prueba para el camino básico #1.

Caso de prueba para el camino básico #1:

Descripción:

Los parámetros de entrada serán listas de objetos de una clase y representan la estructura de la base de datos, las listas no deben estar vacías.

Condición de funcionamiento:

Son especificados los parámetros de entrada para que cumplan una condición de escenario perfecto para el funcionamiento del procedimiento.

Entrada:

Se exponen los parámetros que entran al procedimiento, en este caso se refiere a la subred a escanear.

Resultado Esperado:

Se muestra el resultado que se espera que devuelva el procedimiento.

Resultado:

Se muestra el resultado obtenido por el procedimiento.

Camino Básico #1 : 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 9, 13, 14, 15, 17, 2, 18	
Descripción:	Se escanea una subred y se detecta el servicio Gclient.
Condición de Ejecución:	Debe haberse escogido una subred y una máscara como parámetro.
Entrada:	self.subnet, self.mascara (nombre de la subred y valor de la máscara respectivamente).
Resultados Esperados:	El sistema debe exponer el listado de las PC pertenecientes a la subred que se encuentren encendidas y determinar la situación del servicio Gclient en cada una.
Resultados:	Se muestra el listado con las PC encendidas y la condición del Gclient de forma correcta.

Después de aplicado los casos de prueba, se comprobó que es correcto el flujo de trabajo de la funcionalidad. Se le aplicó este tipo de pruebas a otras funcionalidades del módulo consideradas como importantes para el funcionamiento lo que permite catalogar como válido el código del sistema.

3.7.4 Pruebas de rendimiento

Mediante las pruebas de rendimiento es posible hallar tendencias y comportamientos para los elementos de una aplicación, los cuales generan bajo rendimiento. Este tipo de pruebas permiten identificar cuellos de botella, capacidad de concurrencia de usuarios, tiempos de respuesta de operaciones de negocio a nivel de sistema, establecer un marco de referencia para pruebas futuras, determinar el cumplimiento de los objetivos de rendimiento y requerimientos no funcionales. (31)

Durante la exposición de los resultados se emplean variables como son:

Rendimiento: Número de peticiones procesadas en una unidad de tiempo, que puede ser segundos, minutos y horas.

Min: Tiempo mínimo de conexión entre el total de solicitudes realizadas en milisegundos.

Max: Tiempo máximo de conexión entre el total de solicitudes realizadas en milisegundos.

Error: Representa el error (en porciento) respecto al número total de peticiones.

3.7.4.1 Prueba de Carga

A traves de la ejecución de las pruebas de Carga es posible identificar la capacidad de recuperación de un sistema cuando es sometido a cargas variables tanto de usuarios como de procesos. Al realizar las pruebas de carga se puede determinar el tiempo de respuesta de todas las transacciones críticas del sistema y encontrar cuellos de botella de la aplicación. (29)

Se establece a continuación los elementos a tener en cuenta para la realización de las pruebas:

- ✓ Comportamiento del sistema al recibir 467 peticiones de forma concurrente.
- ✓ Recursos necesarios:

Tipo de prueba	Software	Hardware
Prueba de Carga	Apache JMeter 2.11	- Procesador: Intel(R) Core(TM) i5-4210M 2.4 GHz - RAM: 2GB - Disco Duro: 1TB

Tabla 26. Recursos necesarios para la prueba de carga.

Resultados:

Min: 48

Max: 31072

Rendimiento: 14.4 peticiones/seg.

Error: 0 %

Gráfico del resultado:

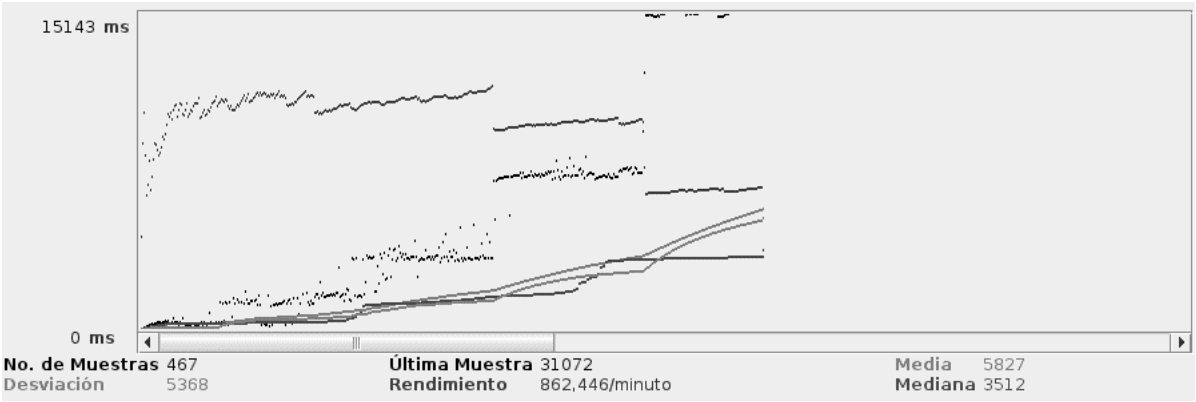


Ilustración 8. Gráfico del resultado de la prueba de carga.

3.7.4.2 Prueba de Estrés

Las pruebas de estrés son útiles para identificar la capacidad de respuesta de un sistema bajo condiciones de carga extrema, representadas por una alta concurrencia de usuarios y/o procesos. Una vez realizadas las pruebas de estrés se podrá conocer el punto de quiebre del aplicativo en términos de capacidad de respuesta, con lo cual será posible establecer acciones de optimización en diferentes niveles para asegurar una mejor capacidad de concurrencia de usuarios y/o procesos que se verá reflejada en una óptima operación de negocio. (29)

A continuación se establecen los elementos a tener en cuenta para la realización de la prueba.

- ✓ Comportamiento del sistema cuando 468 usuarios intentan conectarse concurrentemente al servidor.
- ✓ Recursos necesarios:

Tipo de prueba	Software	Hardware
Prueba de Estrés	Apache JMeter 2.11	- Procesador: Intel(R) Core(TM) i5-4210M 2.4 GHz - RAM: 2GB - Disco Duro: 1TB

Tabla 27. Recursos necesarios para la prueba #2 de estrés.

Resultados:

Min: 27

Max: 31091

Performance: 14.5 peticiones/seg.

Error: 1.71%

Gráfico del resultado:

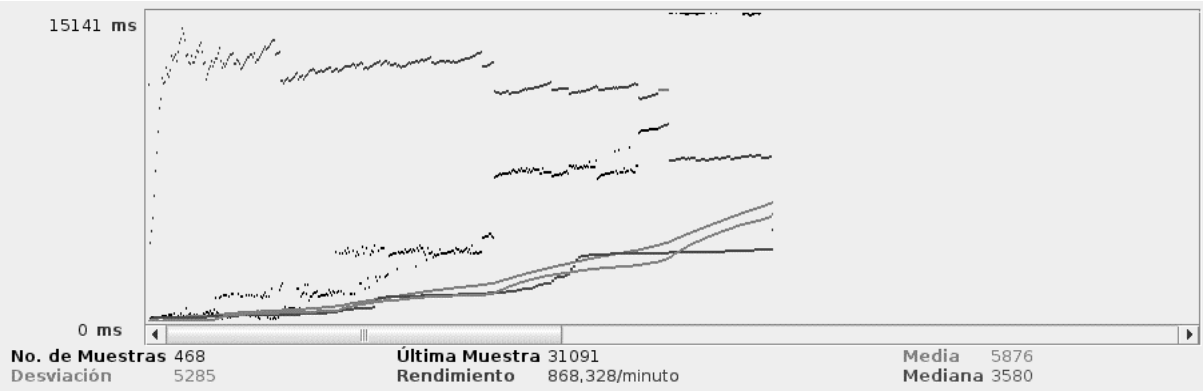


Ilustración 9. Gráfico del resultado de la prueba de carga.

3.7.4.3 Prueba de Desempeño

Posterior a las pruebas antes señaladas, se realizó una prueba teniendo en cuenta el tiempo de ejecución de la funcionalidad de escaneo de subred del módulo anterior para comprobar si el módulo actualizado es capaz de superar al anterior en este parámetro, además del uso de recursos de hardware que representa para el servidor. Según el resultado de dicha prueba, se puede demostrar si es determinante la de una actualización para el sistema existente. (29)

Para ello se utilizó la siguiente configuración de hardware y software:

Hardware	Software
CPU: Intel(R) Core(TM) i5-4210M	Sistema Operativo: Lubuntu 15.04 64 bits
Velocidad CPU: 2.40 GHz	Navegador Web: Mozilla Firefox 45

Memoria RAM: 2GB	
Disco duro: 1 TB	

Tras la ejecución del método Escanear, primero con la implementación del módulo anterior se obtuvieron los siguientes resultados, es necesario tener en cuenta que los valores de uso del CPU y Memoria RAM incluyen el procesamiento natural del sistema operativo:

Numero de subredes a escanear:	1
Uso del CPU (valor aproximado):	27%
Uso de Memoria RAM:	1125 MB
Tiempo de ejecución:	14 minutos y 8 segundos

Al realizar la misma prueba, teniendo en cuenta la misma configuración de hardware y software, pero en este caso con el módulo actualizado, se obtuvo el siguiente resultado:

Numero de subredes a escanear:	5
Uso del CPU (valor aproximado):	27%
Uso de Memoria RAM (valor aproximado):	1125 MB
Tiempo de ejecución:	14 minutos y 10 segundos

Los resultados demuestran una paridad en cuanto a uso de recursos y tiempo de ejecución de cada versión, es evidente que esto representa un aprovechamiento por parte de la nueva versión la cual presenta un rendimiento elevado teniendo en cuenta el tiempo y uso del hardware de la PC.

3.8 Conclusiones parciales

En desarrollo del presente capítulo se identificaron los patrones de diseño GRASP permitiendo que la programación estuviera bien estructurada. Además, facilitó la asignación de responsabilidades logrando un diseño de software que sirva de apoyo a la implementación del sistema. Se ejemplifican algunas de las tarjetas CRC definidas y las tareas de ingeniería, necesarias para establecer el orden de implementación de las HU y del sistema. Se realizaron las pruebas de aceptación, además de las pruebas de caja blanca y las de rendimiento logrando resultados satisfactorios en cuanto al funcionamiento del código, interacción con la interfaz y las mejoras incorporadas con respecto a la versión anterior.

Conclusiones Generales

A partir del desarrollo de la presenta investigación se arribaron a las siguientes conclusiones:

El análisis y estudio de las tecnologías y del módulo anteriormente desarrollado para el sistema GRHS permitió, definir los lenguajes, herramientas y métodos a seguir para la implementación del módulo actual. El desarrollo de la aplicación, dirigido por las actividades establecidas en el marco de trabajo de la metodología XP, logró una organización en el proceso de diseño e implementación del sistema. Se identificaron 6 HU que guiaron la implementación de las funcionalidades deseadas por el cliente. El uso del patrón arquitectónico unido a los patrones GRASP, permitió en la implementación del sistema, agrupar las responsabilidades similares para favorecer la comprensibilidad y cohesión del software. Para comprobar la calidad y correcto funcionamiento del sistema, se diseñaron y ejecutaron pruebas de aceptación, unitarias y de rendimiento, lo que arrojó resultados satisfactorios, demostrando el cumplimiento de los requisitos funcionales establecidos en la fase inicial del proceso de desarrollo del componente.

Recomendaciones

- ✓ Es necesario modificar el módulo de tal forma que el proceso de escaneo detecte las propiedades tales como el identificador del Gclient, aunque este proceso no se encuentre conectado al Gserver que está escaneando.
- ✓ Que se inicie el escaneo automático en el momento exacto que se adicione a la lista de agentes un cliente con dirección IP perteneciente a una subred desconocida.

Referencias Bibliográficas

1. UCI. [En línea] [Citado el: 6 de febrero de 2016.] http://www.uci.cu/?q=mision.

2. NMAP.org. [En línea] https://nmap.org/man/es/.

3. HowtoForge. [En línea] https://www.howtoforge.com/network_monitoring_with_ntop.

4. Calles, Juan Antonio. http://www.flu-project.com. [En línea] miércoles 26 de marzo de 2014. http://www.flu-project.com/2014/03/ejecutando-nmap-desde-python.html..

5. Colombia, Universidad Autónoma de. SlideShare. [En línea] http://es.slideshare.net/yeiko11/protocolo-rip.

6. CISCO. [En línea] 2008 de marzo de 23. http://www.cisco.com/cisco/web/support/LA/7/73/73214_1.html.

7. Herramientas Web. [En línea] [Citado el: 07 de marzo de 2016.] http://neo.lcc.uma.es/evirtual/cdd/tutorial/red/icmp.html.

8. Lagarde, Thierry. AutoScan-Network. [En línea] http://autoscan-network.com/.

9. OPMANTEK. [En línea] [Citado el: 17 de febrero de 2016.] https://opmantek.com/network-discovery-inventory-software/.

10. http://blog.pandorafms.org. [En línea] http://blog.pandorafms.org/herramientas-de-red-tu-kit-para-gestion-de-redes/?lang=es.

11. Universidad de Murcia. [En línea] [Citado el: 17 de febrero de 2016.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

12. Kniberg, Henrik. *SCRUM y XP desde las trincheras.* EUA : Desarrollo de Software Empresarial de InfoQ, 2007. ISBN 978-1-4303-2264-1.

13. Ingenieria de Software. [En línea] [Citado el: 12 de febrero de 2016.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

14. Instituto Tecnológico de Celaya. [En línea] [Citado el: 17 de febrero de 2016.] www.iqcelaya.itc.mx/~vicente/Programacion/Lenguajes.doc..

15. Alvarez, Miguel Angel. desarrolloweb.com. [En línea] 19 de noviembre de 2003. http://www.desarrolloweb.com/articulos/1325.php.

16. SQL, Curso. Sphinx. [En línea] 18 de abril de 2015. http://csrg.inf.utfsm.cl/~jfuentes/_build/html/lectures/week1/lecture1.html..

17. postgresql-es. [En línea] [Citado el: 12 de febrero de 2016.] http://www.postgresql.org.es/sobre_postgresql.

18. Gutiérrez, Javier. [En línea] [Citado el: 17 de febrero de 2016.] www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf..

19. AndresGF. vanuta. [En línea] 2 de septiembre de 2013. http://vanuta.com/articulos/que-es-django-framework/.

20. Docsetools. [En línea] [Citado el: 17 de febrero de 2016.] http://docsetools.com/articulos-para-saber-mas/article_56561.html..

21. andrearrs. hipertextual. [En línea] 14 de 06 de 2014. http://hipertextual.com/archivo/2014/06/pycharm-ide-python/.

22. Guión Visual Paradigm. [En línea] [Citado el: 17 de febrero de 2016.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf..

23. BizAgi, Corporate. BizAgi Process Modeler. [En línea] [Citado el: 17 de febrero de 2016.] http://www.bizagi.com/esp/descargas/BPMNbyExample.pdf..

24. Cornejo, José Enrique González. DocIRS. [En línea] enero de 2008. http://www.docirs.com/uml.htm.

25. Pressman, Roger S. *Ingeniería de Software, un enfoque práctico (5ta Edición).* s.l. : Mc Graw Hill, 2011.

26. Sun Microsystems. *Distributed Applications Architecture.* 2009.

27. M., Saúl García. *La guía definitiva de Django, Desarrolla aplicaciones Web de forma rápida y sencilla.* 2015.

28. Schmuller, Joseph. *Aprendiendo UML en 24 horas.* s.l. : Prentice Hall.

29. Londoño, Jorge Hernan Abad. Ingenieria de Software. [En línea] miercoles 6 de abril de 2005. http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html.

Bibliografía

1. Flu-Project.com [En linea] miercoles, 26 de marzo de 2014 http://www.flu-project.com/2014/03/ejecutando-nmap-desde-python.html

2. NMAP.org [En línea] https://nmap.org

3. *blog.crespo.org.ve* [En línea] 11 de enero de 2015 http://blog.crespo.org.ve/2015/01/descubrir-equipos-en-una-red-con-ipcalc.html

4. Ingeniería de Software [En línea] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html

5. Arlety de la Caridad Sebastian Martínez, René Juán Pérez de Corcho Hernández. *Descubrimiento de la red.* 2015.

6. Herramientas Web. [En línea] [Citado el: 07 de marzo de 2016.] http://neo.lcc.uma.es/evirtual/cdd/tutorial/red/icmp.html.1. UCI. [En línea] [Citado el: 6 de febrero de 2016.] http://www.uci.cu/?q=mision.

2. NMAP.org. [En línea] https://nmap.org/man/es/.

3. HowtoForge. [En línea] https://www.howtoforge.com/network_monitoring_with_ntop.

4. Calles, Juan Antonio. http://www.flu-project.com. [En línea] miércoles 26 de marzo de 2014. http://www.flu-project.com/2014/03/ejecutando-nmap-desde-python.html..

5. Colombia, Universidad Autónoma de. SlideShare. [En línea] http://es.slideshare.net/yeiko11/protocolo-rip.

6. CISCO. [En línea] 2008 de marzo de 23. http://www.cisco.com/cisco/web/support/LA/7/73/73214_1.html.

7. Herramientas Web. [En línea] [Citado el: 07 de marzo de 2016.] http://neo.lcc.uma.es/evirtual/cdd/tutorial/red/icmp.html.

8. Lagarde, Thierry. AutoScan-Network. [En línea] http://autoscan-network.com/.

9. OPMANTEK. [En línea] [Citado el: 17 de febrero de 2016.] https://opmantek.com/network-discovery-inventory-software/.

10. http://blog.pandorafms.org. [En línea] http://blog.pandorafms.org/herramientas-de-red-tu-kit-para-gestion-de-redes/?lang=es.

11. Universidad de Murcia. [En línea] [Citado el: 17 de febrero de 2016.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

12. Kniberg, Henrik. *SCRUM y XP desde las trincheras.* EUA : Desarrollo de Software Empresarial de InfoQ, 2007. ISBN 978-1-4303-2264-1.

13. Ingenieria de Software. [En línea] [Citado el: 12 de febrero de 2016.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

14. Instituto Tecnológico de Celaya. [En línea] [Citado el: 17 de febrero de 2016.] www.iqcelaya.itc.mx/~vicente/Programacion/Lenguajes.doc..

15. Alvarez, Miguel Angel. desarrolloweb.com. [En línea] 19 de noviembre de 2003. http://www.desarrolloweb.com/articulos/1325.php.

16. SQL, Curso. Sphinx. [En línea] 18 de abril de 2015. http://csrg.inf.utfsm.cl/~jfuentes/_build/html/lectures/week1/lecture1.html..

17. postgresql-es. [En línea] [Citado el: 12 de febrero de 2016.] http://www.postgresql.org.es/sobre_postgresql.

18. Gutiérrez, Javier. [En línea] [Citado el: 17 de febrero de 2016.] www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf..

19. AndresGF. vanuta. [En línea] 2 de septiembre de 2013. http://vanuta.com/articulos/que-es-django-framework/.

20. Docsetools. [En línea] [Citado el: 17 de febrero de 2016.] http://docsetools.com/articulos-para-saber-mas/article_56561.html..

21. andrearrs. hipertextual. [En línea] 14 de 06 de 2014. http://hipertextual.com/archivo/2014/06/pycharm-ide-python/.

22. Guión Visual Paradigm. [En línea] [Citado el: 17 de febrero de 2016.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf..

23. BizAgi, Corporate. BizAgi Process Modeler. [En línea] [Citado el: 17 de febrero de 2016.] http://www.bizagi.com/esp/descargas/BPMNbyExample.pdf..

24. Cornejo, José Enrique González. DocIRS. [En línea] enero de 2008. http://www.docirs.com/uml.htm.

25. Pressman, Roger S. *Ingeniería de Software, un enfoque práctico (5ta Edición).* s.l. : Mc Graw Hill, 2011.

26. Sun Microsystems. *Distributed Applications Architecture.* 2009.

27. M., Saúl García. *La guía definitiva de Django, Desarrolla aplicaciones Web de forma rápida y sencilla.* 2015.

28. Schmuller, Joseph. *Aprendiendo UML en 24 horas.* s.l. : Prentice Hall.

29. Londoño, Jorge Hernan Abad. Ingenieria de Software. [En línea] miercoles 6 de abril de 2005. http://ing-sw.blogspot.com/2005/04/tipos-de-pruebas-de-software.html.

30. Williams, Laurie. *White Box Testing.* 2006.

31. Potencier, Fabien y Zaninotto, François. *Synphony, la guía definitiva.* 2007.

32. Fisanotti, Juan Pedro. *Introducción a Django.* 2011.

33. Arlety de la Caridad Sebastian Martínez, René Juán Pérez de Corcho Hernández. *Descubrimiento de la red.* 2015.

34. Foundation, The Apache Software. Apache JMeter. [En línea] 1999. http://jmeter.apache.org/.