Universidad de las Ciencias Informáticas

Facultad 1

Título: Sistema de recopilación del uso de aplicaciones para Nova GNU/Linux.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es):

Carlos Cesar Caballeros Díaz

José Daniel Cañada Castillo

Tutores:

Lic. Vianka Orovio Cobo,

Ing. Amaury Viera Hernández

Ciudad de La Habana, Cuba, mayo 2012.

"Año 54 de la Revolución"

“La elección fácil era unirme al mundo del software propietario, firmar los acuerdos de no revelar, y prometer que no iría en ayuda de mi amigo hacker. Podría haber hecho dinero de esta manera, y tal vez me hubiera divertido escribiendo código. Pero sabía que al final de mi carrera, al mirar atrás los años, la pase construyendo paredes para dividir a la gente, sentiría que usé mi vida para empeorar el mundo.”

Richard M. Stallman.

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos al departamento SIMAYS de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de junio del año 2012.

José Daniel Cañada Castillo

_____

Firma del autor

Carlos Cesar Caballero Díaz

_____

Firma del autor

Lic. Vianka Orovio Cobo

_____

Firma del tutor

Ing. Amaury Viera Hernández

_____

Firma del tutor

Agradecimientos

De Carlos Cesar:

Primeramente a mis padres, a quienes debo todo lo que soy, y a toda mi familia por estar siempre pendientes y brindarme todo su apoyo, colaboración y cariño.

A mis incondicionales amigos del grupo 108, por cargar conmigo durante estos 5 años, en especial Rubén, Ernesto, Novoa, Yasmani y Jorge.

A Lara y Johan, quienes he conocido a lo largo de estos 5 años y me han demostrado su amistad.

A todos los amigos gracias a los cuales hoy escribo estas líneas, necesitaría incontables páginas para agradecerles y saben que no soy bueno escribiendo lo que siento.

A mis tutores Vianka y Amaury, que siempre nos dieron su apoyo y brindaron sus conocimientos.

A Yoandy y Alexander que fueron de gran ayuda para la realización del presente trabajo.

A mi compañero de tesis, por todo el trabajo realizado.

A mis compañeros del proyecto, con quienes he compartido el día a día y siempre han estado dispuestos a tender una mano.

Y en general agradezco a todos lo que de una forma u otra me han ayudado a permanecer estos 5 años, son tantos que es imposible ponerlos a todos, pero saben que no olvido.

De José Daniel:

Le agradezco:

A mi madre, que siempre me ha enseñado a ser el hombre que hoy en día soy, por ser la persona más incansable cuando se trata de dar amor.

A mi abuela, siempre llevo presente, porque es la base de mi formación, desde pequeño aprendí mucho de ella y hoy en día soy también el fruto de su amor. De ese amor que siempre supo darle a todos sus nietos.

A ese gran tío, padre y amigo, ya que siempre supo darme de él todo a cambio de nada, nada más que su amor, su respeto y su sabiduría.

A mi hermano, que siempre supo darme fuerzas para continuar.

A mi hermana y mi sobrina por ese amor de familia que nunca me faltó.

A toda mi familia que de una forma u otra influyeron en mi formación profesional.

A mi compañero de tesis, que es la persona que más respeto por su preocupación y dedicación ante las tareas, fue siempre un bastón donde siempre me pude apoyar sin importar nada.

A mis amigos de siempre.

Agradecimientos

De Carlos Cesar y José Daniel:

A nuestros tutores: Amaury y Vianka. Especialmente a Amaury que estuvo en todo momento y fue nuestro apoyo y nuestra guía.

A nuestros profesores y compañeros de aula.

A todos nuestros compañeros y amigos a lo largo de estos 5 años en la UCI.

A Fidel, a Raúl y a la Revolución Cubana por permitirnos graduarnos como Ingenieros.

A todos muchas gracias.

Dedicatoria

A todos los que de una forma u otra me ayudaron a transitar por mis estudios universitarios.

A mis amigos que siempre me han apoyado.

A mi familia, que siempre ha estado ahí.

Y a mis padres por ser los mejores.

Cesar

Esta tesis está dedicada a mi mamá, a mi tío y padre, a mi hermano, a mi sobrina, a mi hermana.

A toda mi familia tan querida que tengo.

Pero en especial, a mi abuela Esperanza que siempre estará viva en mi recuerdo.

José Daniel

Resumen

El proyecto SIMAYS de la Universidad de las Ciencias Informáticas, es el encargado de brindar soporte técnico a la distribución cubana de GNU/Linux Nova, así como otros servicios relacionados con la migración a software libre en Cuba, pero no cuenta con información precisa del uso que los usuarios le dan a los programas de dicha distribución.

El objetivo central de este trabajo, es el análisis, diseño e implementación de un sistema, que permita recopilar información acerca del uso que dan los usuarios a los repositorios y aplicaciones de la distribución cubana de GNU/Linux Nova, el cual permita facilitar y mejorar el soporte técnico y otros servicios que brinda el proyecto SIMAYS de la Universidad de las Ciencias Informáticas.

Índice de Contenido

Introducción

Las compañías que desarrollan y mantienen sistemas operativos o aplicaciones siempre han tenido la necesidad de acercarse a sus usuarios para obtener información sobre estos programas, con el fin de evaluar la acogida de sus productos para replantear y hacer más productivo y eficiente el soporte técnico, además de otros servicios que brindan. Con el paso del tiempo y el aumento del número de usuarios, se hizo evidente la necesidad de crear sistemas automatizados para la recogida de dicha información, controlando así las cada vez mayores barreras físicas y culturales.

Los sistemas informáticos que recolectan información en los sistemas operativos y aplicaciones son una excelente herramienta para los equipos de desarrollo y soporte técnico, pues brindan la posibilidad de obtener información de primera mano, sin necesidad de invertir demasiado tiempo y recursos en la confección de encuestas ni en la realización de entrevistas; las cuales muchas veces debido al poco conocimiento del personal encuestado o entrevistado conlleva a errores o a cúmulos de información escasamente útiles.

En el caso de las compañías que se dedican al desarrollo de productos informáticos como sistemas operativos y sus aplicaciones agregadas, la información recolectada facilitará la toma de decisiones sobre cuales paquetes de programas deben aparecer en las versiones de sus sistemas operativos. Tal es el caso de la compañía Canonical Ltd., que patrocina el desarrollo de la distribución Ubuntu GNU/Linux, la cual mejora cada nueva entrega del producto, colocando los paquetes más populares entre la comunidad, para que los nuevos usuarios los tengan instalados una vez que obtengan una copia del sistema operativo.

No obstante, los usuarios ofrecen resistencia a su uso debido al desconocimiento del tipo de información que es enviada a los desarrolladores o equipos de soporte, temen que sea enviada información privada, confidencial o sensible. Otro de los mitos que rodean a estos sistemas es la creencia de que constituyen puertas traseras, mediante las cuales se puede acceder deliberadamente a la información sensible o privada de los usuarios, almacenada en los equipos.

La migración a software libre en Cuba constituye una realidad. Este es un proceso llevado a cabo por el Grupo de Capacitación, el Grupo Legal, el Grupo de Divulgación y el Grupo Técnico Nacional. Este último integrado por el Departamento de Sistemas Operativos y Desarrollo de Tecnologías Libres (SODTL) de la Universidad de las Ciencias Informáticas (UCI) y el proyecto Servicios Integrales de Migración, Asesoría y Soporte (SIMAYS), del centro CESOL, en la Universidad de las Ciencias Informáticas. SIMAYS colabora de forma directa con los desarrolladores del sistema operativo cubano Nova GNU/Linux, que es la distribución GNU/Linux propuesta en la Guía Cubana de Migración a Software Libre y Código Abierto(1), brindando los servicios que complementan la migración del país, como asesoría, consultoría, capacitación y soporte técnico.

El sistema operativo Nova GNU/Linux es creado por el proyecto Nova en la Universidad de las Ciencias Informáticas, en el año 2004. En el año 2006 se realizó una primera distribución en su versión 1, por algunas instituciones del país como las FAR, teniendo muy buena aceptación por sus clientes. Actualmente, este sistema operativo se encuentra en su versión 3 y tiene varias líneas como Nova Light, Nova Desktop, Nova Server, entre otros. Como toda distribución de GNU/Linux cuenta con repositorios de aplicaciones informáticas, que garantizan el buen funcionamiento del sistema. La mayoría de estas aplicaciones son instaladas y usadas a conveniencia del usuario, por lo que se desconoce cuáles son las más utilizadas o la de mayor preferencia por ellos.

El proyecto SIMAYS no cuenta con un sistema informático para recopilar información acerca del uso de las aplicaciones en el sistema operativo Nova GNU/Linux lo que trae como consecuencia la inexistencia de información precisa del uso que los usuarios le dan a los programas. Analizando esta situación puede identificarse el siguiente **problema científico**: ¿Cómo recopilar información acerca del uso de aplicaciones en el sistema operativo Nova GNU/Linux?

Para dar solución al mismo se define como **objetivo general** desarrollar un sistema que permita la recopilación de información acerca del uso de las aplicaciones y repositorios en el sistema operativo Nova GNU/Linux. El **objeto de estudio** de la investigación son los sistemas de recopilación de información, y el **campo de acción** la recopilación de información acerca del uso de aplicaciones y repositorios de paquetes.

Se pretende cumplir el **objetivo general** a partir de los siguientes objetivos específicos:

- Analizar los sistemas informáticos que recolectan información del uso de aplicaciones y repositorios de paquetes.
- Identificar las herramientas, metodología de desarrollo de software y tecnologías.
- Analizar y diseñar la solución propuesta.
- Implementar los sistemas que componen la solución propuesta.
- Probar el sistema desarrollado.

Los anteriores objetivos se concretan en las siguientes **tareas de investigación**:

- Revisión de la bibliografía sobre los sistemas de recopilación de información del uso de aplicaciones y repositorios de paquetes.
- Selección de las herramientas, metodología de desarrollo de software y tecnologías a utilizar.
- Análisis del sistema propuesto.
- Descripción detallada de la arquitectura.
- Implementación de las funcionalidades del sistema propuesto.
- Diseño y ejecución de pruebas al sistema implementado

En esta investigación se tiene como **idea a defender** que el desarrollo de un sistema que recopile información del uso de aplicaciones reunirá los datos necesarios acerca de cómo son utilizados los programas en el sistema operativo Nova GNU/Linux.

En el transcurso de la investigación se emplean métodos y técnicas de investigación de gran importancia para el desarrollo del trabajo, utilizándose como método teórico el **analítico-sintético**, que permite el estudio de diferentes fuentes bibliográficas para extraer los elementos más importantes que se relacionan con los sistemas que recopilan información del uso de aplicaciones y repositorios, así como de los repositorios de paquetes que brindan soporte al sistema operativo Nova GNU/Linux. Se realizaron además resúmenes y valoraciones de conceptos relevantes relacionados con estos temas.

Para la recogida de la información, se empleó la técnica tormenta de ideas. La misma fue realizada con especialistas del departamento Migración y Soporte y personas de la comunidad de Software Libre. Donde se obtuvieron diferentes puntos de vista e ideas para la elaboración de la aplicación.

El presente documento se estructura en tres capítulos, los cuales se describen brevemente a continuación:

Capítulo 1: Fundamentación Teórica.

En este capítulo se lleva a cabo un estudio de las herramientas o sistemas de recopilación de información en los ordenadores, específicamente sobre el uso de aplicaciones y descargas de repositorios de paquetes, sus principales funcionalidades, entornos en los que se despliegan los sistemas y los requisitos del software. Además, se describe la metodología a utilizar para el desarrollo del sistema, así como los lenguajes de programación, librerías y herramientas que se utilizan.

Capítulo 2: Análisis y diseño de la solución propuesta.

En este capítulo se muestra la lista de reserva del producto, así como cada una de las Historias de Usuario que tendrá el sistema, analizándose las características para determinar una solución adecuada y conveniente. Se realiza una descripción de la arquitectura y los patrones de diseños aplicados en el sistema. Además se explican los diagramas de clases y de base de datos, entre otros.

Capítulo 3: Implementación y pruebas al sistema.

En este capítulo se describe la implementación del sistema y se muestran los principales resultados obtenidos. Además, se presentan los casos de prueba para las distintas Historias de Usuario así como el estilo de código a emplear.

Capítulo 1. Fundamentación teórica.

Para la realización del trabajo de diploma es necesario conocer el concepto relacionado con repositorios de paquetes y sus características: son dos elementos que guiarán el desarrollo del Sistema de recopilación de uso de aplicaciones para Nova GNU/Linux. Se realiza el estudio de las herramientas caracterizándolas en dos grupos: las que realizan la recopilación de información del uso de aplicaciones en las computadoras y las que realizan la recopilación de información del uso de aplicaciones en los repositorios.

1.1 Repositorios de Paquetes:

En sistemas como Debian que es una distribución GNU/Linux, la mayoría de los software están empaquetados en ficheros .deb (o .rpm, como en Red Hat) que contienen los programas y bibliotecas necesarios. Estos archivos pueden ser descargados desde internet o venir en Cds o DVDs (2).

1.1.1 ¿Qué son los repositorios de paquetes?

Repositorios, depósitos o archivos son sitios centralizados (servidores) donde se almacena y mantiene información digital, habitualmente bases de datos, archivos informáticos o juegos de paquetes (3).

1.1.2 Estructura de un repositorio de paquetes

Un repositorio consiste en al menos un directorio con algunos paquetes deb en él, y dos ficheros especiales que son el Packages.gz para los paquetes binarios y el Sources.gz para los paquetes de las fuentes. Además constituyen una colección de herramientas que sirven para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software (4).

1.1.3 Descargas desde el repositorio de paquetes

Existen dos vías para acceder a estos repositorios para realizar alguna descarga de los paquetes: accediendo a una página web o usando un gestor de paquetes. Esto genera un fichero .log en el servidor Apache que contiene información válida para analizar operaciones que se realizan diariamente en el servidor, por ejemplo: ip, fecha, hora, arquitectura, paquete.

IP:	Ip desde el cual se realizó la descarga del paquete.
Fecha:	Es el día, mes y año exacto de cuando se realizó la descarga.
Arquitectura:	Es la información de la arquitectura que pertenece ese paquete, mayormente i386.
Paquete:	Es el software descargado por su nombre en el repositorio.

1.1.4 Paquetes de repositorios instalados.

Dentro de los conceptos asociados al dominio del problema, están los relacionados a los sistemas de paquetería utilizados en las distintas distribuciones del sistema operativo GNU/Linux, abordando de forma más específica los relacionados con la distribución Nova GNU/Linux. Una vez que el repositorio esté listado correctamente en el sources.list (Si se cumple la sintaxis del sources.list), apt-get realizará las búsquedas especificadas, sean con ficheros binarios o con fuentes en los archivos Packages.gz o Sources.gz respectivamente. Esto se debe a que en el fichero Packages.gz se encuentra toda la información de los paquetes, como nombre, versión, tamaño, descripción corta y larga, las dependencias y alguna información adicional. Sin embargo, en el fichero Sources.gz se encuentran listados todos los nombres, versiones y las dependencias de desarrollo (los paquetes necesitados para compilar) de los paquetes, cuya información es usada por apt-get source u otro gestor de paquetes.

1.2 *Servicios Web*

Los Servicios Web (Web Service) son aplicaciones o tecnologías con capacidad para interoperar en la web. Estas intercambian datos con el objetivo de ofrecer servicios, los cuales ofrecen mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario, para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, para lo que es necesaria una arquitectura de referencia estándar (5).

Un Servicio Web es una entidad ejecutable de software, totalmente encapsulada y "autosuficiente", que puede ser detectada e invocada a través de una red para cumplir con un propósito determinado. El "consumidor" de un Servicio Web desconoce la complejidad interior de la misma, la plataforma y el lenguaje en el que está escrita, pero sí conoce el efecto concreto de su ejecución (6).

Los servicios web utilizan como su gran insumo el lenguaje extensible de marcado XML y se basan en una arquitectura en la que se define el servicio web a través de uno de los lenguajes estándar, se publica en un directorio donde se halla la descripción anteriormente hecha y se utiliza de acuerdo a las expectativas para resolver una necesidad de acuerdo con la descripción provista. La arquitectura que mejor se ha adaptado al mundo de los servicios web es SOA brindando un enfoque que ha adoptado los negocios y ha incrementado el intercambio electrónico de datos y el comercio electrónico (7).

1.2.1 SOA

SOA (Service Oriented Architecture) es un conjunto de patrones, principios y prácticas para construir piezas de software que puedan interoperar independientemente de la tecnología empleada en su implementación. En este sentido, SOA implica la superación de tecnologías que no permiten la interoperabilidad entre distintas tecnologías de desarrollo de software (8).

1.2.2 SOAP

SOAP (Simple Object Access Protocol) es un protocolo ligero para el intercambio de información estructurada en un entorno descentralizado y distribuido. Es un protocolo basado en XML que consta de tres partes: un sobre que define un marco para describir lo que está en un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias de solicitud de tipos de datos definidos, y una convención para representar llamadas a procedimientos remotos y respuestas (9).

1.2.3 Descubrimiento de Servicios

Es la detección automática de dispositivos y servicios que son ofrecidos en una red de computadoras. Para posibilitar el descubrimiento de servicios mediante la red, se utilizan protocolos que brindan la capacidad de identificar automáticamente los servicios en condiciones de cumplir con ciertos requisitos. Utilizando un protocolo de descubrimiento de servicios se puede acceder a un recurso sin necesidad de conocer su localización real (10, 11).

1.3 *Mapeo objeto-relacional*

El mapeo objeto-relacional, más conocido por su nombre y siglas en inglés Object-relational mapping (ORM), es una técnica de programación que permite la integración entre el modelo de datos orientado a objetos y el modelo relacional, lo que posibilita el uso de características propias de la orientación a objetos como la herencia y el polimorfismo. Las bases de datos del tipo relacional solo permiten guardar tipos de datos primitivos, por lo que no es posible guardar objetos de forma directa, sino que estos se deben convertir antes en registros, que por lo general afectan a varias tablas. En el momento de volver a recuperar los datos, hay que hacer el proceso contrario, se deben convertir los registros en objetos. En este punto es que se muestra la importancia del ORM, ya que este se encarga de forma automática de convertir los objetos en registros y viceversa, simulando así tener una base de datos orientada a objetos (12). Otra de las ventajas del mapeo objeto-relacional, es la portabilidad que ofrecen los frameworks, ya que dan la posibilidad de adaptar su sistema a múltiples bases de datos distintas sin necesidad de alterar el sistema en modo alguno.

1.4 *Análisis de sistemas de recopilación de información de uso de aplicaciones y descargas en repositorios de paquetes existentes.*

Después de una exhaustiva investigación en los diferentes motores de búsquedas y meta-buscadores existentes en la red, así como en parte de la Web Invisible haciendo uso de las bases de datos a las cuales tiene acceso la UCI, se puede indicar que no existe ningún sistema desarrollado que incluya la recopilación de información del uso de aplicaciones y las descargas de repositorios de paquetes, existiendo algunos sistemas que analizan el uso de aplicaciones y otros que recopilan información de los repositorios, pero ninguno que realice la tarea de forma conjunta. Por esta razón se decide extender la búsqueda a sistemas que recopilen información del uso de aplicaciones y a sistemas que recopilen información de las descargas en

repositorios de paquetes, para entonces realizar un análisis de las principales características y funcionalidades de las mismas.

1.4.1 Análisis de sistemas de recopilación de información de uso de aplicaciones.

1.4.1.1 *Herramientas del software original de Microsoft.*

Las herramientas del programa software de Microsoft original (llamado "Programa de Ventajas de Windows Original" en Windows XP y "Tecnologías de activación de Windows" en Windows Vista y Windows 7) son un conjunto de herramientas que se encargan de recopilar y analizar información acerca de la configuración, el estado del equipo y las aplicaciones de Microsoft, para controlar la piratería de software, analizando entre otras cosas si las aplicaciones son utilizadas de forma ilegal.

Características:

- Determina si se está ejecutando una copia con licencia apropiada de software de Microsoft.
- Detecta, y desactiva, las "vulnerabilidades de la activación".
- Recopila la siguiente información:
- Marca y modelo del equipo.
- Información de la versión del sistema operativo y del software.
- Configuración regional y de idioma.
- Un número exclusivo que las herramientas asignan al equipo (Identificador global único o GUID).
- Clave del producto (con hash) e identificador del producto.
- Nombre, número de revisión y fecha de revisión del BIOS.
- Número de serie del disco duro (con hash).
- Si la instalación se completó correctamente, en caso de que se llevara a cabo.
- El resultado de la comprobación de validación, en el que se incluye información y códigos de error de todas las vulnerabilidades de activación y de cualquier software malintencionado o no autorizado que se haya encontrado o deshabilitado, incluido lo siguiente:
    - Identificador de la vulnerabilidad de activación.
    - Estado actual de la vulnerabilidad de activación como por ejemplo, limpiada o en cuarentena.
    - Identificación del fabricante del equipo Original.
    - Nombre del archivo de la vulnerabilidad de activación y hash del archivo, así como un hash de los componentes del software relacionados que puedan indicar la presencia de una vulnerabilidad de activación.

- Nombre y contenido del archivo de instrucciones de arranque del equipo (conocido como el archivo de arranque), que ayuda a detectar las vulnerabilidades de activación que modifican este archivo.
- La dirección de protocolo de Internet (IP) se registra temporalmente cuando el equipo se conecta a un servidor o a un sitio web software original de Microsoft.

- Las herramientas que conforman el programa Software de Microsoft Original están licenciadas bajo la EULA de Microsoft.

1.4.1.2 *Smolt*

Smolt versión 1.1.1.1-56.1, es una base de perfiles de hardware desarrollado por el proyecto Fedora que permite que sus usuarios envíen los detalles de su configuración durante la instalación. Smolt, como PackageKit, es independiente de la distribución y su instalación es opcional. A pesar de que OpenSUSE ha incluido a Smolt en sus repositorios desde hace un tiempo, finalmente ahora tomaron la gran decisión de integrarla en su propio instalador.

Representa el trabajo conjunto de pocos proyectos de Linux para obtener información sobre el hardware de los equipos que ejecutan Linux como sistema operativo. Está configurado para no ejecutarse de forma automática, por lo que se debe iniciar manualmente. Cuando el perfil de hardware se ha completado, el programa presentará la información recogida y solicita el permiso de envío al servidor de Smolt. La información se recopila en una base de datos central anónimamente garantizando su privacidad.

La recopilación de información de hardware en un sitio centralizado es un deseo que existe desde hace mucho tiempo en GNU/Linux. Smolt no es el primer intento, ni siquiera el único que se está ejecutando actualmente, pero es la primera iniciativa que es aceptada por algunas distribuciones de GNU/Linux.

Características:

Recopila datos como una iniciativa entre proyectos de distintas distribuciones, dando como resultado una gran cantidad de información que se puede utilizar para:

- Ayudar a los desarrolladores en la detección de hardware que no esté bien soportado.
- Centrar los esfuerzos en el hardware más popular.
- Ofrecer alternativas y soluciones mediante consejos en la wiki de Smolt o, en el caso de OpenSUSE, consejos específicos, que se brindan en la sección de hardware de su portal.
- Ayudar a los usuarios a elegir la distribución que mejor se adapte a su hardware.
- Convencer a los fabricantes de hardware para que den soporte a sus equipos en GNU/Linux y no sólo en otros sistemas operativos mayoritarios.

Este software está incluido por defecto en OpenSUSE desde la versión 11.1, actualmente se encuentra en su versión smolt-1.1.1.1-56.1 (13, 14).

1.4.1.3 *Debian Popularity Contest*

El "Debian Popularity Contest" o "Concurso de Popularidad de Debian" es un proyecto que mapea el uso de los paquetes de la distribución Debian GNU/Linux. Cuenta con un sitio web que publica las estadísticas recolectadas por el paquete popularity-contest, el cual se instala a petición del usuario en el proceso de instalación, y envía semanalmente información acerca de los paquetes instalados al servidor que procesa y publica la información.

Características:

- Recoge fecha y hora en la que se realiza un envío.
- Recoge un listado de todos los paquetes instalados en el sistema.
- Recoge la fecha y hora en que se instaló o actualizó por última vez un paquete.
- Recoge la fecha y hora en que se utilizó por última vez un paquete.
- Recoge la arquitectura del sistema.
- Marca los paquetes indicando:
  - Si se usa de forma regular.
  - Si se usa de forma regular.
  - Si no se usa de forma regular
  - Si se ha actualizado el paquete recientemente.
  - Si el paquete no contiene ninguna información.
- Utiliza una arquitectura cliente-servidor.
- Cuenta con una página web que muestra las estadísticas recogidas por el proyecto.
- Realiza los envíos de la información recogida mediante http o correo electrónico.
- Utiliza un estilo de "concurso" que da puntuación a los paquetes utilizados con más frecuencia.
- Para la recolección y envío de datos en los clientes utiliza un script programado en Perl.
- Licencia GPL.

Actualmente se encuentra en la versión 1.54.

1.4.1.4 *Ubuntu Popularity Contest*

Proyecto gemelo al Debian Popularity Contest, siendo sus principales diferencias:

- El paquete popularity-contest está instalado por defecto en la distribución Ubuntu GNU/Linux, aunque su uso queda a elección del usuario.
- Cuenta con un sitio web para mostrar las estadísticas mantenido por el Ubuntu Web Team.

Actualmente se encuentra en la versión 1.48.

1.4.2 Análisis de sistemas de recopilación de información de uso de repositorios.

1.4.2.1 *OpenSUSE Statistics*

Proyecto oficial de la distribución OpenSUSE que analiza las descargas a sus repositorios oficiales para recopilar información, siendo sus principales características:

- Cuenta con un sitio que muestra estadísticas del uso de la distribución.
- Los datos son recopilados completamente desde los logs de Apache del sitio download.opensuse.org.

Como no existe referencia de la versión de OpenSUSE Statistics, se toma la del sistema (OpenSUSE versión 12.1).

1.5 *Herramientas, lenguajes y metodología a utilizar.*

1.5.1 Herramientas de desarrollo

1.5.1.1 *Netbeans IDE*

Netbeans versión 7.1, es un entorno de desarrollo integrado libre para desarrolladores de software. Es un proyecto exitoso con una gran base de usuarios, una comunidad en constante crecimiento, está escrito en java, pero soporta una gran cantidad de lenguajes de programación como el propio java, C++, PHP, JavaScript y Groovy. Existe además un número importante de módulos para extender el Netbeans, como su módulo para Python, haciéndolo uno de los mejores IDE para desarrollar en ese lenguaje (15, 16).

1.5.1.2 *Eclipse*

Eclipse versión 3.6.7, es un entorno de desarrollo integrado de código abierto, multiplataforma; es desarrollado por la Fundación Eclipse, que es una organización sin fines de lucro y ayuda a cultivar tanto una comunidad de código abierto como un ecosistema de productos y servicios complementarios. El Proyecto Eclipse fue creado originalmente por IBM en noviembre del 2001 y apoyado por un consorcio de proveedores de software. La Fundación Elipse (Eclipse Foundation) se creó en enero de 2004 como una organización independiente sin fines de lucro. Fue creada para permitir que un proveedor neutral, abierto y transparente se estableciera alrededor de Eclipse y su comunidad. Hoy en día, la comunidad Eclipse se compone de individuos y organizaciones que son una muestra representativa de la industria del software (17).

Eclipse cuenta con un desarrollo muy activo, y una buena cantidad de módulos que lo hacen una excelente herramienta para el desarrollo, con soporte actualmente para una gran variedad de lenguajes de programación como Java, Python, Perl, PHP y muchos otros.

1.5.1.3 *MySQL Workbeanch*

MySQL Workbeanch versión 5.2, es una herramienta gráfica para trabajar con servidores de bases de datos MySQL, permite la creación y administración de conexiones a los servidores de bases de datos. Contiene un editor SQL que facilita realizar consultas a la base de datos, permitiendo crear de forma gráfica modelos de la base de datos y crearlos utilizando ingeniería inversa, además de editar todos los aspectos de la base de datos mediante un editor de tablas; también permite crear y administrar instancias de los servidores. Está disponible en dos ediciones, la Comunity Edition y la Estandard Edition la cual provee características empresariales.

1.5.2 Herramientas para la gestión de datos

1.5.2.1 *MySQL*

MySQL versión 5.1, se ha convertido en uno de los gestores de bases de datos más populares del mundo Open Source debido a que es fácil de usar y tiene un muy buen rendimiento. Muchas de las grandes compañías desarrolladoras de software como Facebook, Google o Adobe utilizan este sistema de bases de datos para sus servicios ahorrando tiempo y dinero. MySQL se ejecuta en más de 20 plataformas incluyendo Linux, Windows, Mac OS, Solaris y IBM AIX (18).

1.5.2.2 *Pyws*

Pyws versión 1.1.1, es un proyecto cuyo propósito es ayudar a los desarrolladores a exponer funciones de sus sistemas como APIs públicas vía SOAP. La idea principal es permitir un trabajo rápido y sin complicaciones, sin necesidad de centrarse en SOAP, generando de forma dinámica las descripciones WSDL.

Principales características:

- Framework web completamente independiente del servidor.
- Contiene un adaptador para Django.
- Soporta los protocolos: SOAP 1.1, REST, JSON.
- Permite manipulación de contexto y contiene un framework de autenticación.
- Contiene un sistema de descripción para tipos de datos simples.

Características SOAP:

- Solicitud, respuesta, excepciones.
- Descripción del servicio WSDL 1.1, incluyendo las cabeceras y las excepciones.

- Manejo de tipos de datos simples: enteros, flotantes, cadenas, fecha, fecha y hora.
- Manejo de tipos de datos complejos: diccionarios y listas (permite estructuras anidadas de cualquier profundidad).
- Pruebas de integración: PHP, Java (Axis 1.4), WS-I Basic Profile 1.2.

1.5.2.3 *Storm*

Storm versión 0.15, es un mapeador objeto-relacional (ORM) para Python desarrollado por Canonical y es usado en proyectos como Launchpad y Landscape. Está diseñado para trabajar tanto con pequeñas como grandes bases de datos soportando SQLite, PostgreSQL y MySQL. Tiene un buen diseño y una API limpia y ligera que ofrece una curva de aprendizaje corta y mantenimiento a largo plazo. Storm es rápido, robusto, bien documentado y libre, estando licenciado bajo la LGPL 2.1 (19).

1.5.2.4 *Minusconf*

Minusconf versión 1.0, es un protocolo de localización de servicios implementado en Python, es ligero y permite múltiples servicios por máquina sin necesidad de configuraciones entre servicios, tampoco requiere de configuraciones para su funcionamiento a diferencia de otros similares. Su principal característica es que es muy simple y carece de descripción de dispositivos y subscripciones.

1.5.3 Lenguajes de programación

1.5.3.1 *Python*

Python versión 2.6, es un lenguaje script independiente de la plataforma y orientado a objetos, preparado para realizar cualquier tipo de programas, 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 (20). Algunos casos de éxito en el uso de Python son Google, Yahoo, la NASA, Industrias Light & Magic, y todas las distribuciones Linux, en las que Python cada vez representa un tanto por ciento mayor de los programas disponibles (21).

1.5.3.2 *PERL*

Perl versión estable 5.16.0, es un lenguaje de programación interpretado, al igual que muchos otros lenguajes de Internet como JavaScript o ASP. Esto quiere decir que el código de los scripts en Perl no se compila sino que cada vez que se quiere ejecutar se lee el código y se pone en marcha interpretando lo que hay escrito. Además es extensible a partir de otros lenguajes, ya que desde Perl se pueden hacer llamadas a subprogramas escritos en otros lenguajes. También desde otros lenguajes es posible ejecutar código Perl (22).

1.5.4 Metodología ágil a utilizar

Dadas las características del proyecto a desarrollar, donde las entregas funcionales deben realizarse en poco tiempo y de una forma continua, se decide la utilización de una metodología ágil de desarrollo de software,

lo cual permite una fácil y rápida adaptabilidad ante cualquier cambio, logrando una respuesta versátil al cambio y maximizando de esta forma los beneficios.

Entre las metodologías ágiles más conocidas se encuentran:

- XP (Extreme Programming).
- Feature-Driven Development (FDD).
- Dynamic Systems Development Method (DSDM).
- SCRUM.
- Adaptive Software Development (ASD).
- Metodología Crystal.
- AUP.
- SXP.

Para el desarrollo de este trabajo se propone el uso de SXP.

1.5.4.1 SXP

Es una metodología compuesta por las metodologías SCRUM y XP, la cual ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles, que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva, fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo.

SCRUM es una forma de gestionar un equipo de manera que se trabaje de forma eficiente y de tener siempre medidos los progresos.

XP más bien es una metodología encaminada para el desarrollo; consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto.

Consta de 4 fases principales:

- Planificación: Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- Desarrollo: Es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
- Entrega: Puesta en marcha.
- Mantenimiento: Donde se realiza el soporte para el cliente.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos.

1.6 Conclusiones parciales

En el desarrollo de este capítulo se abordaron conceptos generales acerca de los repositorios, así como los sistemas existentes que recopilan información sobre el uso de aplicaciones en los distintos sistemas operativos, con el objetivo de ubicar al lector en el dominio del problema. Fueron analizados algunos de los sistemas de recopilación información sobre el uso de aplicaciones, para documentar las principales características que proveen además se determinó que es necesaria la implementación de un sistema que recopile información ya que las existentes no cumplen con las expectativas del cliente.

Para ello, se abordan elementos de la metodología SXP que guiará el proceso de desarrollo de software, así como las principales tecnologías y lenguajes usados en la construcción de aplicaciones web dinámicas, seleccionando MySQL 5.1 como gestor de base de datos y Python 2.6 para el desarrollo, y como Sistema Operativo la distribución GNU/Linux Nova 3.0.

Capítulo 2. Análisis y diseño de la solución propuesta.

En el presente capítulo se propone una solución que cumple con las expectativas del cliente. Se describen los requisitos del sistema, tanto funcionales como no funcionales, en la lista de reserva de producto (LRP), además de las historias de usuario correspondientes a los requisitos funcionales del sistema. Se elaboran los diagramas de componentes del sistema, los diagramas de clases de los subsistemas así como el diagrama de diseño de la base de datos.

2.1 *Propuesta del sistema a desarrollar*

Para dar solución al problema planteado, se propone el desarrollo de un sistema que conste de:

- Una aplicación cliente que se encargue de monitorear las descargas realizadas en los repositorios de software.
- Una aplicación cliente que se encargue de analizar el uso de aplicaciones en estaciones de trabajo.
- Una aplicación servidor, que reúna la información obtenida por las aplicaciones clientes en una base de datos y permita mediante servicios web acceder a los datos necesarios.

2.2 *Planificación por roles*

Rol	**Responsabilidad**	**Nombre**
Gerente	Dirige y controla las tareas del equipo. Toma las decisiones finales. Participa en la selección de objetivos y requerimientos. Controla el progreso y da seguimiento a cada iteración. Evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes.	Amaury Viera Hernández Vianka Orovio Cobo
Cliente	Participa en las tareas que involucran la lista de reserva del producto.	SIMAYS, UCI
Programador	Elabora el código de las nuevas funcionalidades a implementar. Escribe las pruebas unitarias. Debe existir una comunicación y coordinación adecuada entre los programadores y el resto del equipo.	Carlos Cesar Caballero
Analista	Es el encargado de escribir las Historias de	José Daniel Cañada

	Usuario y las pruebas funcionales para validar su implementación.	
Diseñadores	Encargado del diseño del sistema, de los prototipos de interfaces máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción	Carlos Cesar Caballero José Daniel Cañada
Encargado de pruebas	Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Carlos Cesar Caballero José Daniel Cañada
Arquitecto	Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas.	José Daniel Cañada Carlos Cesar Caballero

2.3 *Lista Reserva de Producto (LRP)*

La LRP es una lista priorizada que define el orden por importancia del trabajo que se va a realizar en el desarrollo del proyecto. En las fases de inicio, es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, los más importantes suelen surgir en los comienzos del ciclo de desarrollo y por lo general son suficientes para una primera iteración. Siempre el objetivo es asegurar que el producto definido al terminar sea el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto.

Para el éxito de la aplicación se debe tener en cuenta cual operación realizar primero, es decir, que tenga un mayor impacto o influencia en el funcionamiento del sistema. Para ello se definen las prioridades de las funciones que debe realizar el sistema, entre la actualización constantemente de las operaciones realizadas en el servidor y las computadoras clientes, enviando esos datos al servidor para su almacenamiento (23).

La lista de reserva del producto, estará en constante cambio a medida que se vaya profundizando en el sistema hasta su completa implementación o por nuevas peticiones del cliente agregando nuevas funcionalidades, dejando aperturas para nuevas versiones con mejoras en su implementación. Se deja escrito correctamente los requisitos no funcionales que aunque no sean de gran impacto para la aplicación, se deben tener en cuenta para el soporte del sistema. Una buena documentación de estos requisitos no funcionales garantiza que en el futuro, en caso de mantenimiento del sistema, el tiempo que se requiera sea el mínimo o de menor costo.

2.3.1 RF (Requisitos Funcionales)

RF (Requisitos Funcionales)				
Asignado a	**Ítem**	**Descripción**	**Estimación**	**Estimado por**
Prioridad		***Alta***		
Carlos Cesar José Daniel	**1**	Asignar identificador al sistema. (cliente de estación de trabajo).	1 día	Analista y Programador
	2	Asignar dirección estática del servidor a utilizar. (cliente de estación de trabajo)	1 día	Analista y Programador
	3	Asignar grupo. (cliente de estación de trabajo)	1 día	Analista y Programador
	4	Obtener identificador del sistema. (cliente de estación de trabajo)	1 día	Analista y Programador
	5	Obtener arquitectura del sistema. (cliente de estación de trabajo)	1 día	Analista y Programador
	6	Obtener listado de paquetes instalados en el sistema.	1 día	Analista y Programador
	7	Obtener hora de instalación o actualización de los paquetes instalados en el sistema.	1 día	Analista y Programador
	8	Obtener hora del último uso de los paquetes instalados en el sistema.	1 día	Analista y Programador
	9	Obtener hora de ejecución del chequeo de uso de paquetes.	1 día	Analista y Programador
	10	Marcar los paquetes que han sido instalados o actualizados recientemente.	1 día	Analista y Programador
	11	Marcar los paquetes que no han sido usados en más de 30 días (un mes).	1 día	Analista y Programador
	12	Marcar los paquetes que no contienen ficheros (metapaquetes o paquetes virtuales).	1 día	Analista y Programador

	13	Marcar los paquetes que han sido usados en menos de 30 días.	1 día	Analista y Programador
	14	Establecer conexión de forma estática con el servidor. (cliente de estación de trabajo)	1 día	Analista y Programador
	15	Enviar reporte al servidor. (Cliente de estación de trabajo).	1 día	Analista y Programador
	16	Asignar identificador al sistema. (cliente de repositorios)	1 día	Analista y Programador
	17	Asignar dirección estática del servidor a utilizar. (cliente de repositorios)	1 día	Analista y Programador
	18	Asignar grupo. (cliente de repositorios)	1 día	Analista y Programador
	19	Asignar ruta de la raíz del host virtual del servidor Apache.	1 día	Analista y Programador
	20	Asignar ruta de la raíz del repositorio en el servidor Apache.	1 día	Analista y Programador
	21	Asignar distribución del repositorio.	1 día	Analista y Programador
	22	Obtener identificador del sistema cliente estación de trabajo.	1 día	Analista y Programador
	23	Obtener lista de paquetes del repositorio descargados.	1 día	Analista y Programador
	24	Obtener IP desde donde fueron descargados los paquetes del repositorio.	1 día	Analista y Programador
	25	Obtener fecha y hora en la que fueron descargados los paquetes del repositorio.	1 día	Analista y Programador
	26	Obtener versión de los paquetes del repositorio descargados.	1 día	Analista y Programador

	27	Obtener arquitectura de los paquetes del repositorio descargados.	1 día	Analista y Programador
	28	Obtener distribución de los paquetes descargados.	1 día	Analista y Programador
	29	Establecer conexión de forma estática con el servidor. (cliente de repositorios)	1 día	Analista y Programador
	30	Enviar reporte al servidor. (cliente de repositorios)	1 día	Analista y Programador
	31	Crear Base de datos.	1 día	Analista y Programador
	32	Recibir reportes de clientes de estación de trabajo.	1 día	Analista y Programador
	33	Recibir reportes de los clientes de repositorios.	1 día	Analista y Programador
	34	Devolver cantidad de descargas dado un paquete.	1 día	Analista y Programador
	35	Devolver la puntuación dado un paquete.	1 día	Analista y Programador
Prioridad		***Media***		
	36	Asignar la configuración de la dirección del servidor a utilizar en el cliente de la estación de trabajo. (dinámica o estática)	1 día	Analista y Programador
	37	Establecer conexión de forma dinámica con el servidor. (cliente de estación de trabajo)	1 día	Analista y Programador
	38	Asignar configuración de dirección del servidor a utilizar en el cliente de repositorios. (dinámica o estática)	1 día	Analista y Programador

	39	Establecer conexión de forma dinámica con el servidor. (cliente de repositorios)	1 día	Analista y Programador

2.3.2 RNF (Requisitos No Funcionales)

RF (Requisitos Funcionales)				
Asignado a	**Ítem**	**Descripción**	**Estimación**	**Estimado por**
	Prioridad	***Baja***		
Carlos Cesar José Daniel	1	Utilizar como servidor web Apache y servidor de bases de datos MYSQL.		Analista y Programador
	2	El sistema deberá funcionar sobre el sistema operativo GNU/Linux Nova.		Analista y Programador
	3	Disponer de los siguientes paquetes: - Python 2.6 o 2.7 - python-cherrypy3		Analista y Programador
	4	Disponer de los siguientes paquetes en la aplicación servidor: - python-storm		Analista y Programador
	5	Disponer del siguiente paquete en la aplicación cliente de estaciones de trabajo: - Perl		Analista y Programador
	6	Usar como lenguaje de programación Python y Perl.		Analista y Programador

2.4 *Historia de usuario (HU)*

Las Historias de Usuario son la técnica utilizada en SXP para especificar los requisitos de la aplicación, equivalente a los casos de uso en el proceso unificado de desarrollo (RUP). Son las tareas y funciones que el software debe hacer, escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto.

Las Historias de Usuario guían la construcción de las pruebas de aceptación, en las cuales debe generarse una o más pruebas para verificar que han sido correctamente realizadas y son utilizadas para estimar tiempos de desarrollo.

A continuación se relacionan las Historias de Usuario; se vinculan con la prioridad que tienen y los usuarios encargados de desarrollarlas. El proceso varía en dependencia de las especificaciones que el usuario realiza durante el desarrollo del sistema. Es decir, todas las decisiones, recomendaciones y sugerencias que impliquen cambios en el desarrollo del software se toman de conjunto con el cliente.

Historia de Usuario HU-01

Historia de Usuario	
Número: HU-01	**Nombre Historia de Usuario**: Asignar identificador al sistema. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Cuando el sistema se instala en el cliente de estación de trabajo, genera un identificador único, para enviar los reportes al servidor; de esta forma el servidor identifica que cliente está enviando la información a almacenar.	
Observaciones: Esta operación se hace automáticamente cuando se instala la aplicación.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-02

Historia de Usuario	
Número: HU-02	**Nombre Historia de Usuario**: Asignar dirección estática del servidor a utilizar. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Se asigna una dirección estática del servidor a la aplicación.	
Observaciones: Se especifica la dirección del servidor que se va utilizar para almacenar los datos. Se configura una vez instalada la aplicación a través de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-03

Historia de Usuario	
Número: HU-03	**Nombre Historia de Usuario**: Asignar grupo. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Asignará el grupo al que pertenece el cliente de repositorios.	
Observaciones: Se realiza una vez que está instalada la aplicación, se configura el grupo al que pertenece. Esta configuración se realiza a través de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-04

Historia de Usuario	
Número: HU-04	**Nombre Historia de Usuario**: Obtener identificador del sistema. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtener el identificador del sistema que el servidor utilizará para reconocer el cliente.	
Observaciones: Este identificador se recoge de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-05

Historia de Usuario	
Número: HU-05	**Nombre Historia de Usuario**: Obtener arquitectura del sistema. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtener la arquitectura del sistema operativo donde está ejecutándose la aplicación.	
Observaciones:	
Prototipo interfaz: No posee.	

Historia de Usuario HU-06

Historia de Usuario	
Número: HU-06	**Nombre Historia de Usuario**: Obtener listado de paquetes instalados en el sistema.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtener el listado de los paquetes instalados en el sistema operativo donde está ejecutándose la aplicación.	
Observaciones:	
Prototipo interfaz: No posee.	

Historia de Usuario HU-07

Historia de Usuario	
Número: HU-07	**Nombre Historia de Usuario**: Obtener hora de instalación o actualización de los paquetes instalados en el sistema.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene la fecha y hora en que se instaló o actualizó por última vez cada uno de los paquetes instalados en el sistema operativo.	
Observaciones:	
Prototipo interfaz: No posee.	

Historia de Usuario HU-08

Historia de Usuario	
Número: HU-08	**Nombre Historia de Usuario**: Obtener hora del último uso de los paquetes instalados en el sistema.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene la fecha y hora en que se utilizó por última vez cada uno de los paquetes instalados en el sistema operativo.	
Observaciones:	
Prototipo interfaz: No posee.	

Historia de Usuario HU-09

Historia de Usuario	
Número: HU-09	**Nombre Historia de Usuario**: Obtener hora de ejecución del chequeo de uso de paquetes.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene la fecha y hora en que se efectuó el chequeo de uso de paquetes.	
Observaciones:	
Prototipo interfaz: No posee.	

Historia de Usuario HU-10

Historia de Usuario	
Número: HU-10	**Nombre Historia de Usuario**: Marcar los paquetes que han sido instalados o actualizados recientemente.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Marca los paquetes que han sido instalados o actualizados recientemente en el sistema operativo.	
Observaciones: La instalación o actualización se considera reciente cuando se ha realizado en un plazo menor a un mes.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-11

Historia de Usuario	
Número: HU-11	**Nombre Historia de Usuario**: Marcar los paquetes que no han sido usados en más de 30 días (un mes).
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Marca los paquetes que no se han usado en los últimos 30 días.	
Observaciones: En el reporte se agrega una entrada con la etiqueta <OLD> que indica si un paquete no ha sido usado en los últimos 30 días.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-12

Historia de Usuario	
Número: HU-12	**Nombre Historia de Usuario**: Marcar los paquetes que no contienen ficheros (metapaquetes o paquetes virtuales).
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Marca los paquetes que no contienen ficheros.	
Observaciones: En el reporte se agrega una entrada con la etiqueta <NOFILES> a los paquetes que no contienen ficheros. Estos son paquetes virtuales o metapaquetes que usa el sistema operativo.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-13

Historia de Usuario	
Número: HU-13	**Nombre Historia de Usuario**: Marcar los paquetes que han sido usados en menos de 30 días.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Marca los paquetes que han sido usados en los últimos 30 días.	
Observaciones: En el reporte se agrega una entrada con la etiqueta <POINT> a los paquetes que han sido usados al menos una vez en los últimos 30 días.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-14

Historia de Usuario	
Número: HU-14	**Nombre Historia de Usuario**: Establecer conexión de forma estática con el servidor. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: El cliente utiliza una dirección estática para establecer la conexión con el servidor.	
Observaciones: La dirección se obtiene de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-15

Historia de Usuario	
Número: HU-15	**Nombre Historia de Usuario**: Enviar reporte al servidor. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: El cliente de estación de trabajo envía un reporte al servidor.	
Observaciones: El reporte contiene una cabecera con información referente a: tipo de reporte, identificador del sistema que envía el reporte, distribución, versión, nombre de la versión, versión de la aplicación cliente, grupo y arquitectura; además un listado de los paquetes instalados conteniendo: fecha y hora de instalación o actualización, fecha y hora de último uso, nombre del paquete, versión, sección y marca.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-16

Historia de Usuario	
Número: HU-16	**Nombre Historia de Usuario**: Asignar identificador al sistema. (cliente de repositorios)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Cuando el sistema se instala en el cliente de repositorios, genera un identificador único, para enviar los reportes al servidor; de esta forma se identifica qué cliente está enviando la información a almacenar.	
Observaciones: Esta operación se hace automáticamente cuando se instala la aplicación.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-17

Historia de Usuario	
Número: HU-17	**Nombre Historia de Usuario**: Asignar dirección estática del servidor a utilizar. (cliente de repositorios)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Se asigna una dirección estática del servidor a la aplicación.	
Observaciones: Se especifica la dirección del servidor que se va utilizar para almacenar los datos. Se configura una vez instalada la aplicación a través de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-18

Historia de Usuario	
Número: HU-18	**Nombre Historia de Usuario**: Asignar grupo. (cliente de repositorios)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Asignará el grupo al que pertenece el cliente de repositorios.	
Observaciones: Se realiza una vez que está instalada la aplicación, se configura el grupo al que pertenece. Esta configuración se realiza a través de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-19

Historia de Usuario	
Número: HU-19	**Nombre Historia de Usuario**: Asignar ruta de la raíz del host virtual del servidor Apache.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Asignará la ruta de la raíz del host virtual del servidor Apache donde están alojados los repositorios.	
Observaciones: El usuario especificará la dirección en un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-20

Historia de Usuario	
Número: HU-20	**Nombre Historia de Usuario**: Asignar ruta de la raíz del repositorio en el servidor Apache.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Asignará la ruta de la raíz del repositorio dentro del host virtual.	
Observaciones: El usuario especificará la dirección en un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-21

Historia de Usuario	
Número: HU-21	**Nombre Historia de Usuario**: Asignar distribución del repositorio.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Asignará la distribución del repositorio.	
Observaciones: El usuario especificará la distribución del repositorio en un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-22

Historia de Usuario	
Número: HU-22	**Nombre Historia de Usuario**: Obtener identificador del sistema cliente estación de trabajo.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtener el identificador del sistema el cual el servidor utilizará para reconocer el cliente.	
Observaciones: Este identificador se recoge de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-23

Historia de Usuario	
Número: HU-23	**Nombre Historia de Usuario**: Obtener lista de paquetes descargados del repositorio.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene cada uno de los paquetes descargados del repositorio.	
Observaciones: Este listado se utiliza para generar el reporte a enviar.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-24

Historia de Usuario	
Número: HU-24	**Nombre Historia de Usuario**: Obtener IP desde donde fueron descargados los paquetes del repositorio.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene los ip desde donde fueron descargados cada uno de los paquetes.	
Observaciones: Se utiliza para generar el reporte a enviar.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-25

Historia de Usuario	
Número: HU-25	**Nombre Historia de Usuario**: Obtener fecha y hora en la que fueron descargados los paquetes del repositorio.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene la fecha y hora en la que fue descargado cada uno de los paquetes.	
Observaciones: Se utiliza para generar el reporte a enviar.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-26

Historia de Usuario	
Número: HU-26	**Nombre Historia de Usuario**: Obtener versión de los paquetes del repositorio descargados.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene la versión de cada uno de los paquetes que son descargados.	
Observaciones: Se utiliza para generar el reporte a enviar.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-27

Historia de Usuario	
Número: HU-27	**Nombre Historia de Usuario**: Obtener arquitectura de los paquetes del repositorio descargados.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene la arquitectura de cada uno de los paquetes que son descargados.	
Observaciones: Se utiliza para generar el reporte a enviar.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-28

Historia de Usuario	
Número: HU-28	**Nombre Historia de Usuario**: Obtener distribución de los paquetes descargados.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Obtiene la distribución de cada uno de los paquetes que son descargados.	
Observaciones: Se utiliza para generar el reporte a enviar.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-29

Historia de Usuario	
Número: HU-29	**Nombre Historia de Usuario**: Establecer conexión de forma estática con el servidor. (cliente de repositorios)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: El cliente utiliza una dirección estática para establecer la conexión con el servidor.	
Observaciones: La dirección se obtiene de un fichero de configuración.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-30

Historia de Usuario	
Número: HU-30	**Nombre Historia de Usuario**: Enviar reporte al servidor. (cliente de repositorios)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: El cliente de estación de trabajo envía un reporte al servidor.	
Observaciones: El reporte contiene una cabecera con información referente a: tipo de reporte, identificador del sistema que envía el reporte, distribución, versión, nombre de la versión, versión de la aplicación cliente y grupo, además un listado de los paquetes descargados conteniendo: ip desde donde ocurre la descarga, fecha y hora de la descarga, nombre del paquete, versión del paquete, sección del paquete y distribución del paquete.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-31

Historia de Usuario	
Número: HU-31	**Nombre Historia de Usuario**: Crear Base de datos.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Crea la base de datos que se va utilizar para guardar los reportes.	
Observaciones: Permite la creación de base de datos MySQL.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-32

Historia de Usuario	
Número: HU-32	**Nombre Historia de Usuario**: Recibir reportes de clientes de estación de trabajo.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Recoger los reportes de los clientes.	
Observaciones: El servidor al recibir los reportes valida la información.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-33

Historia de Usuario	
Número: HU-33	**Nombre Historia de Usuario**: Recibir reportes de los clientes de repositorios.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Recoger los reportes de los clientes.	
Observaciones: El servidor al recibir los reportes valida la información.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-34

Historia de Usuario	
Número: HU-34	**Nombre Historia de Usuario**: Devolver cantidad de descargas dado un paquete.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Devuelve la cantidad de descargas dado un paquete.	
Observaciones: Mediante un servicio web se brinda la cantidad de descargas de un paquete dado.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-35

Historia de Usuario	
Número: HU-35	**Nombre Historia de Usuario**: Devolver la puntuación dado un paquete.
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Dado un paquete devuelve el uso del mismo.	
Observaciones: Mediante un servicio web se brinda información sobre la puntuación de un paquete dado.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-36

Historia de Usuario	
Número: HU-36	**Nombre Historia de Usuario**: Asignar la configuración de la dirección del servidor a utilizar en el cliente de estación de trabajo. (dinámica o estática)
Modificación de Historia de Usuario número: Ninguna.	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 2
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Especificará si la conexión al servidor será de forma estática o dinámica.	
Observaciones: Se especifica en un fichero de configuración si se va a realizar de forma dinámica.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-37

Historia de Usuario	
Número: HU-37	**Nombre Historia de Usuario**: Establecer conexión de forma dinámica con el servidor. (cliente de estación de trabajo)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 1
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Establece la conexión con del servidor de forma dinámica.	
Observaciones: Si no logra establecer la conexión de forma dinámica intenta la conexión de forma estática.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-38

Historia de Usuario	
Número: HU-38	**Nombre Historia de Usuario**: Asignar la configuración de la dirección del servidor a utilizar en el cliente de repositorios. (dinámica o estática)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 2
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Especificará si la conexión al servidor será de forma estática o dinámica.	
Observaciones: Se especifica en un fichero de configuración si se va a realizar de forma dinámica.	
Prototipo interfaz: No posee.	

Historia de Usuario HU-39

Historia de Usuario	
Número: HU-39	**Nombre Historia de Usuario**: Establecer conexión de forma dinámica con el servidor. (cliente de repositorios)
Modificación de Historia de Usuario número: Ninguna	
Usuario: Carlos Cesar Caballero	**Iteración Asignada:** 2
Prioridad del negocio: Alta	**Puntos estimados:** 1 día
Riesgo en desarrollo: Alto	**Puntos reales:** 1 día
Descripción: Establece la conexión con del servidor de forma dinámica.	
Observaciones: Si no logra establecer la conexión de forma dinámica intenta la conexión de forma estática.	
Prototipo interfaz: No posee.	

2.5 *Plan de release*

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	En esta iteración se desarrollarán las Historias de Usuario que tienen prioridad alta.		5 semanas
2	En esta iteración se desarrollarán las Historias de Usuario de media prioridad.		4 semanas
3	En esta iteración se desarrollarán las Historias de Usuario de prioridad baja y se irán integrando con las ya realizadas.		1 semana

2.6 *Estilo arquitectónico cliente–servidor*

El estilo arquitectónico cliente-servidor presente en el sistema en desarrollo consiste en una aplicación servidor y dos aplicaciones clientes, los cuales se comunican entre sí a través de la red, donde el servidor puede servir a varios clientes a la vez de forma concurrente. La arquitectura cliente-servidor es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible, permitiendo la recolección de información en los clientes, luego el procesamiento y almacenamiento de esta en el servidor.

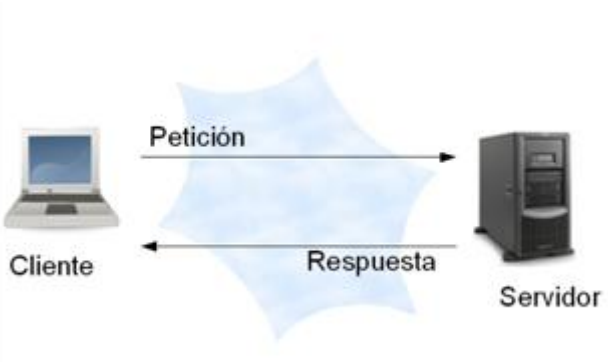


Figura 1. Arquitectura cliente-servidor (23).

2.7 *Diagrama de componentes*

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Estos pueden representar ficheros, bibliotecas, paquetes, librerías, etc. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable.

El sistema de recopilación del uso de aplicaciones está compuesto por tres aplicaciones (ver anexo 1):

- **Cliente de estación de trabajo** (ver anexo 2)
- **Cliente de Repositorio** (ver anexo 3)
- **Servidor** (ver anexo 4)

2.8 *Diagrama de clases del diseño*

El diagrama de clases del diseño es la representación gráfica de las clases que serán implementadas en el sistema. Este diagrama muestra las especificaciones y detalles más concretos de cada clase del sistema así como su relación de asociación, composición o agregación existentes entre ellas.

- **Diagrama de clases del diseño del Subsistema Cliente de Repositorio (ver anexo 5)**
- **Diagrama de clases del diseño del Subsistema Cliente de Estación de trabajo (ver anexo 6)**
- **Diagrama de clases del diseño del Subsistema Servidor (ver anexo 7)**

2.9 *Diseño de la base de datos*

Al ejecutarse los sistemas clientes se inician un conjunto de procesos encargados del monitoreo de las aplicaciones que usan los usuarios y los paquetes descargados de los repositorios. Estos procesos generan una cantidad importante de información la cual necesita ser procesada de una forma rápida y sencilla por el servidor, el cual procesa esta información enviada por las aplicaciones cliente para su almacenamiento.

Para la gestión de esta información se utilizó como sistema gestor de bases de datos MySQL por sus características. A continuación se presenta la descripción del modelo de datos empleado para cada servicio perteneciente al sistema de recopilación del uso de aplicaciones para Nova GNU/Linux.

- ***Diseño de la base de datos*** (ver anexo 8)

2.10 *Conclusiones parciales*

En este capítulo se identificaron los requerimientos que el sistema debe cumplir desglosados en 39 Historias de Usuario y estableciéndose un orden de prioridad para su realización, además de ser asignadas a un desarrollador, para que lleve a cabo una serie de tareas o actividades para que su cumplimiento. Se definió el estilo arquitectónico que se empleó para el diseño del sistema completo. El estilo cliente – servidor es el más indicado para las funciones que realizarán los subsistemas para su interoperabilidad. Se describió una base de datos robusta por la cantidad de información que va a almacenar semanalmente, garantizando que esta información sea la más actualizada para su uso posterior.

Capítulo 3. Implementación y pruebas.

Este capítulo aborda los elementos relacionado con el estándar de código que se tuvo en cuenta para la implementación del sistema además de las pruebas realizadas a los sistemas, a los principales funcionalidades del sistema.

Las pruebas que se le realizan a un software son vitales para ganar en la seguridad y la calidad del mismo, además de encontrar errores previos a la entrega del sistema al usuario final, de modo que una vez entregado al cliente, el producto posea el menor número de inconvenientes posibles.

Es recomendado realizar pruebas al software desde el mismo momento en que se implementa una nueva funcionalidad, logrando así un entorno de desarrollo satisfactorio.

Existen varios tipos de pruebas que pueden ser realizadas a un sistema para así lograr la obtención de un producto funcional y fiable. Sin embargo, si se trata de desarrollo ágil con pruebas constantes, es interesante destacar la utilización de herramientas que automatizan esta actividad.

La automatización de pruebas es uno de los mayores avances en la programación desde la invención de la orientación a objetos. Concretamente en el desarrollo de las aplicaciones web, las pruebas aseguran la calidad de la aplicación incluso cuando el desarrollo de nuevas versiones es muy activo.

3.1 *Estándares de codificación*

Un estándar de codificación constituye un conjunto de reglas que se siguen para la escritura del código fuente. De tal manera que a otros programadores se les facilite entender el código (identificar las variables, las funciones o métodos, etc.). Los estándares de codificación elevan la mantenibilidad del código, sirven como punto de referencia para los programadores, mantienen un estilo de programación y ayudan a mejorar el proceso de codificación, haciéndolo entre otras cosas más eficientes. Para el desarrollo del sistema se siguen las convenciones de código del núcleo de Python PEP 8 (24) con algunas variaciones.

3.2 *Estándar de código para el desarrollo del Sistema de Recopilación del uso de aplicaciones para Nova GNU/Linux*

3.2.1 Nomenclatura de clases y métodos

Los nombres de clases y sus métodos deben estar escritos en minúscula. Las múltiples palabras deben estar separadas por un guión bajo. Ejemplo:

```
def clase:

def metodo_de_ejemplo():
```

3.2.2 Nomenclatura de variables

Las variables deben contener solo minúsculas y si contienen múltiples palabras estar separadas por un guión bajo. Ejemplo:

variable = 0

variable_de_ejemplo = 0

3.2.3 Identación

Usa 4 espacios para cada nivel de identación, nunca serán mezcladas con tabulares. Ejemplo:

```
def metodo():
    if otro_nivel:
        print “identación”
```

3.2.4 Comentarios

Los comentarios del código serán escritos en inglés y deberán ser frases completas.

3.2.4.1 *Comentarios de bloque*

Los comentarios de bloque generalmente se aplican al código que se encuentra a continuación, y se indentan al mismo nivel que dicho código. Cada línea de un comentario de bloque comienza con un # y un único espacio (a menos que haya texto identado dentro del comentario).

Los párrafos dentro de un comentario de bloque se separan por una línea conteniendo un único #.

3.2.4.2 *Cadenas de documentación (docstrings)*

- Se deben escribir docstrings para todas las funciones y métodos públicos. Los docstrings no son necesarios para métodos no públicos, pero deben tener un comentario que describa lo que hace el método. Este comentario debería aparecer antes de la línea "def".

- Es importante notar, que el """ que finaliza un docstring de varias líneas debería situarse en una línea separada, y preferiblemente precederse por una línea en blanco, por ejemplo:

```
“””
Esto es un docstring
“””
```

- Para cadenas de documentación de una línea, es adecuado mantener el """ de cierre en la misma línea.

3.2.5 Codificación de caracteres

El código debería utilizar las codificaciones ASCII o UTF-8.

Los archivos que usan UTF-8 deberían tener una línea de especificación del juego de caracteres.

3.3 *Casos de prueba*

Con el objetivo principal de garantizar la calidad del producto, entre iteración e iteración se definieron un conjunto de casos de prueba de aceptación para poder avanzar hacia una iteración superior. A continuación se relacionan los casos de prueba realizados a varias Historias de Usuario.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-01-01	**Nombre Historia de Usuario:** Asignar identificador al sistema. (cliente de estación de trabajo)
Nombre de la persona que realiza la prueba: Carlos Cesar Caballero Díaz	
Descripción de la Prueba: Se prueba el instalador de la aplicación (paquete .deb).	
Condiciones de Ejecución: • Debe existir un usuario con permisos de administración en el sistema.	
Entrada / Pasos de ejecución: 1. Ejecutar el instalador de paquetes.	
Resultado Esperado: Se crea el archivo de configuración con un identificador asignado.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-04-01	**Nombre Historia de Usuario:** Obtener identificador del sistema. (cliente de estación de trabajo)
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Se obtiene el identificador del sistema, el cual el servidor utilizará para reconocer el cliente.	
Condiciones de Ejecución: • Esté ejecutándose el cron en el sistema.	
Entrada / Pasos de ejecución: 1. Cron inicia el script de ejecución.	
Resultado Esperado: Recoge el identificador del archivo de configuración y lo coloca en el reporte que genera.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-05-01	**Nombre Historia de Usuario:** Obtener arquitectura del sistema. (cliente de estación de trabajo)
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Se obtiene la arquitectura del sistema operativo donde está ejecutándose la aplicación.	
Condiciones de Ejecución: • Esté ejecutándose el cron en el sistema.	
Entrada / Pasos de ejecución: 1. Cron inicia el script de ejecución.	
Resultado Esperado: Recoge la arquitectura del sistema y lo coloca en el reporte que genera.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-06-01	**Nombre Historia de Usuario:** Obtener listado de paquetes instalados en el sistema.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Se obtienen el listado de los paquetes instalados en el sistema operativo donde está ejecutándose la aplicación.	
Condiciones de Ejecución: • Esté ejecutándose el cron en el sistema.	
Entrada / Pasos de ejecución: 1. Cron inicia el script de ejecución.	
Resultado Esperado: Obtiene los paquetes instalados en el sistema y lo coloca en el reporte que genera.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-07-01	**Nombre Historia de Usuario:** Obtener hora de instalación o actualización de los paquetes instalados en el sistema.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene la fecha y hora en que se instaló o actualizó por última vez cada uno de los paquetes instalados en el sistema operativo.	
Condiciones de Ejecución: • Esté ejecutándose el cron en el sistema.	
Entrada / Pasos de ejecución: 1. Cron inicia el script de ejecución.	
Resultado Esperado: Obtiene la fecha y hora del sistema y lo coloca en el reporte que genera.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-08-01	**Nombre Historia de Usuario:** Obtener hora del último uso de los paquetes instalados en el sistema.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene la fecha y hora en que se utilizó por última vez cada uno de los paquetes instalados en el sistema operativo.	
Condiciones de Ejecución: • Esté ejecutándose el cron en el sistema.	
Entrada / Pasos de ejecución: 1. Cron inicia el script de ejecución.	
Resultado Esperado: Obtiene la fecha y hora del último uso de los paquetes del sistema operativo y lo coloca en el reporte que genera.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-09-01	**Nombre Historia de Usuario:** Obtener hora de ejecución del chequeo de uso de paquetes.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene la fecha y hora en que se efectuó el chequeo de uso de paquetes.	
Condiciones de Ejecución: • Esté ejecutándose el cron en el sistema.	
Entrada / Pasos de ejecución: 1. Cron inicia el script de ejecución.	
Resultado Esperado: Obtiene la fecha y hora en que se efectuó el chequeo de uso de paquetes del sistema operativo y lo coloca en el reporte que genera.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-15-01	**Nombre Historia de Usuario:** Enviar reporte al servidor. (Cliente de estación de trabajo).
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: El cliente de estación de trabajo envía un reporte al servidor después de haber recopilado la toda la información referente a los paquetes instalados y usados por los usuarios.	
Condiciones de Ejecución: • Esté ejecutándose el cron en el sistema.	
Entrada / Pasos de ejecución: 1. Cron inicia el script de ejecución.	
Resultado Esperado: Envía el reporte generado al servidor.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-16-01	**Nombre Historia de Usuario:** Asignar identificador al sistema. (cliente de repositorios)
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Se prueba el instalador de la aplicación (paquete .deb)	
Condiciones de Ejecución: • Debe existir un usuario con permisos de administración en el sistema.	
Entrada / Pasos de ejecución: **1.** Ejecutar el instalador de paquetes.	
Resultado Esperado: Se crea el archivo de configuración con un identificador asignado.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-23-01	**Nombre Historia de Usuario:** Obtener lista de paquetes descargados del repositorio.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene cada uno de los paquetes descargados del repositorio.	
Condiciones de Ejecución: • Debe estar instalado el servidor Apache. • Debe haber un repositorio Configurado en el servidor Apache.	
Entrada / Pasos de ejecución: 1. Se inicia el servicio Cliente de repositorios. 2. El Cliente de repositorios comprueba que ha pasado el tiempo definido para envío.	
Resultado Esperado: Obtiene el listado de los paquetes descargados del repositorio y lo coloca en el reporte que genera.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-26-01	**Nombre Historia de Usuario:** Obtener fecha y hora en la que fueron descargados los paquetes del repositorio.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene la fecha y hora en la que fue descargado cada uno de los paquetes desde que se instaló el sistema operativo.	
Condiciones de Ejecución: • Debe estar instalado el servidor Apache. • Debe haber un repositorio Configurado en el servidor Apache.	
Entrada / Pasos de ejecución: 1. Se inicia el servicio Cliente de repositorios. 2. Se descarga un paquete de un repositorio hosteado en el servidor Apache.	
Resultado Esperado: Guarda la fecha y hora de la descarga del paquete en el archivo de datos.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-26-01	**Nombre Historia de Usuario:** Obtener versión de los paquetes del repositorio descargados.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene la versión de cada uno de los paquetes que son descargados del repositorio.	
Condiciones de Ejecución: • Debe estar instalado el servidor Apache. • Debe haber un repositorio Configurado en el servidor Apache.	
Entrada / Pasos de ejecución: 1. Se inicia el servicio Cliente de repositorios. 2. Se descarga un paquete de un repositorio hosteado en el servidor Apache.	
Resultado Esperado: Guarda la versión del paquete descargado en el archivo de datos.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-27-01	**Nombre Historia de Usuario:** Obtener arquitectura de los paquetes del repositorio descargados.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene la arquitectura de cada uno de los paquetes que son descargados del repositorio.	
Condiciones de Ejecución: • Debe estar instalado el servidor Apache. • Debe haber un repositorio Configurado en el servidor Apache.	
Entrada / Pasos de ejecución: 1. Se inicia el servicio Cliente de repositorios. 2. Se descarga un paquete de un repositorio hosteado en el servidor Apache.	
Resultado Esperado: Guarda la arquitectura de los paquetes descargado en el archivo de datos.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-28-01	**Nombre Historia de Usuario:** Obtener distribución de los paquetes descargados.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Obtiene la distribución de cada uno de los paquetes que son descargados del repositorio.	
Condiciones de Ejecución: • Debe estar instalado el servidor Apache. • Debe haber un repositorio Configurado en el servidor Apache.	
Entrada / Pasos de ejecución: 1. Se inicia el servicio Cliente de repositorios. 2. Se descarga un paquete de un repositorio hosteado en el servidor Apache.	
Resultado Esperado: Guarda la distribución del paquete descargado en el archivo de datos.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-32-01	**Nombre Historia de Usuario:** Recibir reportes de clientes de estación de trabajo.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: El servidor obtiene un reporte enviado por un cliente de estación de trabajo con la información referente a los paquetes instalados y usados por los usuarios.	
Condiciones de Ejecución: • Debe estar en ejecución el servidor.	
Entrada / Pasos de ejecución: 1. Se envía un reporte desde una aplicación cliente de estación de trabajo.	
Resultado Esperado: Obtiene el reporte de la aplicación cliente de estación de trabajo y lo almacena en la base de datos.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-33-01	**Nombre Historia de Usuario:** Recibir reportes de los clientes de repositorios.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: El servidor obtiene un reporte enviado por un cliente de repositorios de los paquetes que fueron descargados en los últimos 7 días.	
Condiciones de Ejecución: • Debe estar en ejecución el servidor.	
Entrada / Pasos de ejecución: 1. Se envía un reporte desde una aplicación cliente de repositorios.	
Resultado Esperado: Obtiene el reporte de la aplicación cliente de repositorios y lo almacena en la base de datos.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-34-01	**Nombre Historia de Usuario:** Devolver cantidad de descargas dado un paquete.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Recoger los reportes a través de un servicio web.	
Condiciones de Ejecución: • Debe estar en ejecución el servidor.	
Entrada / Pasos de ejecución: 1. Con el cliente SOAP SUDS se realiza una petición al servicio pasando por parámetros una lista que contiene los nombres de los paquetes.	
Resultado Esperado: Obtiene la cantidad de descargas de los paquetes ordenados por distribución.	
Evaluación de la Prueba: Satisfactoria.	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU-35-01	**Nombre Historia de Usuario:** Devolver uso dado un paquete.
Nombre de la persona que realiza la prueba: José Daniel Cañada Castillo	
Descripción de la Prueba: Dado un paquete devolver el uso del mismo por el usuario desde que se instaló por primera vez.	
Condiciones de Ejecución: • Debe estar en ejecución el servidor.	
Entrada / Pasos de ejecución: 1. Con el cliente SOAP SUDS se realiza una petición al servicio pasando por parámetros una lista que contiene los nombres de los paquetes.	
Resultado Esperado: Obtiene el uso de los paquetes ordenados por distribución.	
Evaluación de la Prueba: Satisfactoria	

3.4 *Conclusiones parciales*

Al implementar los subsistemas en el lenguaje de programación Python se utilizó como estándar de código el mismo que utiliza este lenguaje, con las mismas características que describe Barry en su guía de estilo de Python.

Mediante las pruebas funcionales que fueron realizadas al producto se facilita el proceso de desarrollo de una manera ágil y fiable, influyendo en la rápida corrección de errores en las funcionalidades implementadas. Garantizando la calidad y eficiencia de la aplicación realizada, cumpliendo con el levantamiento de requisitos.

Conclusiones

Una vez realizada la fundamentación teórica que sustentó este trabajo, definidas las características del sistema, efectuada la implementación y la validación del mismo, se obtuvieron resultados que les permite a los autores presentar las siguientes conclusiones:

El estudio realizado a las herramientas existentes a nivel mundial para la recolección de información del uso de aplicaciones y repositorios de paquetes y el análisis de su funcionamiento y sus características fundamentales reveló la inexistencia de un sistema informático que se adapte a las necesidades del proyecto SIMAYS.

Para la construcción del sistema se utilizaron las herramientas, tecnologías y tendencias más actuales, que por sus características particulares se ajustaban a los requerimientos y propósitos deseados.

Para implementar el Sistema de recopilación del uso de aplicaciones para Nova GNU/Linux debieron ser agrupadas las funcionalidades en tres aplicaciones, un cliente de estaciones de trabajo, un cliente de servidor de repositorios y un servidor.

Después de haber probado el Sistema de recopilación del uso de aplicaciones para Nova GNU/Linux se concluye que el sistema está listo para ser utilizado en entornos reales.

En resumen, los principales aportes de este trabajo son:

- Base teórica para el desarrollo de sistemas afines.
- Base para el desarrollo de diferentes análisis estadísticos acerca del uso que dan los usuarios a las aplicaciones en el sistema operativo Nova GNU/Linux.

Los elementos planteados anteriormente permiten afirmar que se cumplió el objetivo de desarrollar un sistema que permita la recopilación de información acerca del uso de las aplicaciones y repositorios en el sistema operativo Nova GNU/Linux.

Recomendaciones

Para dar continuidad a este trabajo se recomienda:

- Agregar al sistema nuevas funcionalidades para la gestión de la base de datos que permita eliminar información desactualizada.

- Utilizar la información que brinda la aplicación para realizar un análisis estadístico para la toma de decisiones de las aplicaciones instaladas por defecto en el sistema operativo Nova GNU/Linux.

Glosario de siglas y términos:

1. Synaptic:

Estas herramientas pueden listar todos los paquetes que se han instalado y los paquetes que están disponibles en los repositorios que se han configurado. También permite realizar búsquedas a los paquetes de los repositorios.

Esta herramienta proporciona un método simple y centralizado de instalación de software y ofrece a los distribuidores una manera centralizada de enviar las actualizaciones de software.

2. apt-get

El comando apt-get es una poderosa herramienta que se usa mediante la línea de comandos y realizan funciones tales como la instalación de nuevos paquetes de software, actualización de paquetes de software existentes, del índice de la lista de paquetes, e incluso la mejora de la todo el sistema.

3. Paquetes

Un paquete es un conjunto de ficheros relacionados con una aplicación. Constituyen la forma básica en que las distribuciones del sistema operativo GNU/Linux distribuyen las aplicaciones.

4. OpenSUSE

OpenSUSE es un sistema operativo libre basado en GNU/Linux. Mediante él se puede navegar en la Web, gestionar el correo electrónico y fotos, hacer trabajo de oficina, reproducir vídeos o música. Actualmente está en su versión 12.1 (25).

5. Log

Archivo que registra movimientos y actividades de un determinado programa. En un servidor web, se encarga de guardar todos los requerimientos y servicios entregados desde él, por lo que es la base del software de estadísticas de visitas (26).

6. Sources.gz

En el fichero Sources.gz se encuentran listados todos los nombres, versiones y las dependencias de desarrollo (los paquetes necesitados para compilar) de todos los paquetes, cuya información es usada por apt-get source o herramientas similares.

7. Packages.gz

En el fichero Packages.gz se encuentra toda la información de todos los paquetes, como nombre, versión, tamaño, descripción corta y larga, las dependencias y alguna información adicional que no es de interés para el trabajo. Toda la información es listada y usada por los Administradores de Paquetes del sistema tales como dselect o aptitude.

8. Licencia EULA

Conocida también como el "End-User License Agreement", es un tipo de licencia utilizada en la mayoría de las aplicaciones comerciales, la cual funciona como un contrato entre el manufacturero o autor y el usuario final. La EULA mayormente desglosa las restricciones de uso de la aplicación (algunos ejemplos son no modificar el código y no compartir la aplicación).

9. Repositorios

Un repositorio, depósito o archivo es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos. Estos depósitos son usados por usuarios a través de la red desde cualquier sitio.

Referencias Bibliográficas

1. VILLAZON, Yoandy Pérez. *Guía cubana para la migración a Software Libre* [online]. May 2008. S.l.: s.n. [Accessed 24 September 2011]. Available from: ftp://ftp.cult.cu/softwarelibre/documentacion/Guia%20cubana%200.32.pdf.

2. Repositorios. In: [online]. [Accessed 13 December 2011]. Available from: http://doc.ubuntu-es.org/Repositorios.

3. Repositorios digitales. In: [online]. [Accessed 11 June 2012]. Available from: http://bibliotecabiologia.usal.es/tutoriales/catalogos-repositorios-bibliosvirtuales/repositorios_digitales.html.

4. LUNAR, Santiago. Los Repositorios. In: [online]. 14 November 2005. [Accessed 13 December 2011]. Available from: http://blog.milmazz.com.ve/archivos/2005/11/14/los-repositorios#definicion.

5. WORLD WIDE WEB CONSORTIUM. Guía Breve de Servicios Web. In: [online]. [Accessed 6 December 2011]. Available from: http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb.

6. TEKNODA. ESA y Web Services en SAP Netweaver. In: [online]. [Accessed 6 December 2011]. Available from: http://www.teknodatips.com.ar/sap-netweaver/32-esa-y-web-services-en-sap-netweaver-introduccion.html.

7. MACHUCA, Carlos Aandrés Morales. Estado del Arte: Servicios Web. In: [online]. [Accessed 6 December 2011]. Available from: http://sites.google.com/site/camoralesma/articulo2.pdf?attredirects=0.

8. VÁZQUEZ, Juan José. ¿Por qué SOA? In: [online]. [Accessed 6 December 2011]. Available from: http://blogs.tecsisa.com/articulos-tecnicos/por-que-soa/.

9. WORLD WIDE WEB CONSORTIUM. SOAP Specifications. In: [online]. [Accessed 6 December 2011]. Available from: http://www.w3.org/TR/soap/.

10. BERNERS-LEE, Tim, HENDLER, James and LASSILA, Ora. The Semantic Web. In: *Scientific American* [online]. 17 May 2001, [Accessed 7 December 2011]. Available from: http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

11. LIEGENER, Benedikt. Service Discovery. In: [online]. [Accessed 7 December 2011]. Available from: http://www.s-cube-network.eu/km/terms/s/service-discovery.

12. BORJA, Angie Alexandra. Mapeo de objeto relacional. In: [online]. [Accessed 13 December 2011]. Available from: http://www.slideshare.net/inspirateunaula/mapeo-de-objeto-relacional.

13. NOVELL, INC. Smolt - openSUSE. In: [online]. [Accessed 13 December 2011]. Available from: http://es.opensuse.org/Smolt.

14. Sitio Oficial de Smolt. In: [online]. [Accessed 13 December 2011]. Available from: http://www.smolts.org/.

15. Python - NetBeans Wiki. In: [online]. 3 December 2010. [Accessed 13 December 2011]. Available from: http://wiki.netbeans.org/Python.

16. ORACLE CORPORATION. Welcome to NetBeans. In: [online]. [Accessed 13 December 2011]. Available from: http://netbeans.org/.

17. THE ECLIPSE FOUNDATION. About the Eclipse Foundation. In: [online]. [Accessed 13 December 2011]. Available from: http://www.eclipse.org/org/.

18. ORACLE CORPORATION. Why MySQL? In: [online]. [Accessed 13 December 2011]. Available from: http://www.mysql.com/why-mysql/.

19. CANONICAL. Storm. In: [online]. 15 March 2011. [Accessed 14 December 2011]. Available from: https://storm.canonical.com/.

20. ALVAREZ, Miguel Angel. Qué es Python. In: [online]. 19 November 2003. [Accessed 13 December 2011]. Available from: http://www.desarrolloweb.com/articulos/1325.php.

21. GONZÁLEZ DUQUE, Raúl. *Python para todos*. S.l.: s.n., [no date].

22. ALVAREZ, Miguel Angel. Qué es Perl. In: [online]. 29 September 2001. [Accessed 13 December 2011]. Available from: http://www.desarrolloweb.com/articulos/541.php.

23. SÁNCHEZ, José Ernesto Torres and MARÍN, Marcos Bagarotti. *Módulo para el control y reporte del uso de dispositivos externos y de aplicaciones en estaciones clientes de competencias caribeñas al estilo ACM-ICPC*. S.l.: s.n., 2011.

24. ROSSUM, Guido van and WARSAW, Barry. PEP 8 -- Style Guide for Python Code. In: [online]. [Accessed 1 June 2012]. Available from: http://www.python.org/dev/peps/pep-0008/.

25. NOVELL, INC. openSUSE.org. In: [online]. [Accessed 13 December 2011]. Available from: http://www.opensuse.org/es/.

26. ARCHIVO MM. Significado y definición de Log. In: [online]. 12 February 2005. [Accessed 13 December 2011]. Available from: http://www.mastermagazine.info/termino/5610.php.

Bibliografía

PEÑALVER ROMERO, Gladys Marsi and MENESES ABAD, Abel. SXP, metodología ágil para proyectos de software libre. 2009.

STALLMAN, Richard M. *Software libre para una sociedad libre* [online]. S.l.: Traficantes de Sueños. [Accessed 7 December 2011]. Available from: http://traficantes.net/index.php/content/download/18110/185232/file/softlibre%20enriquecido.pdf.

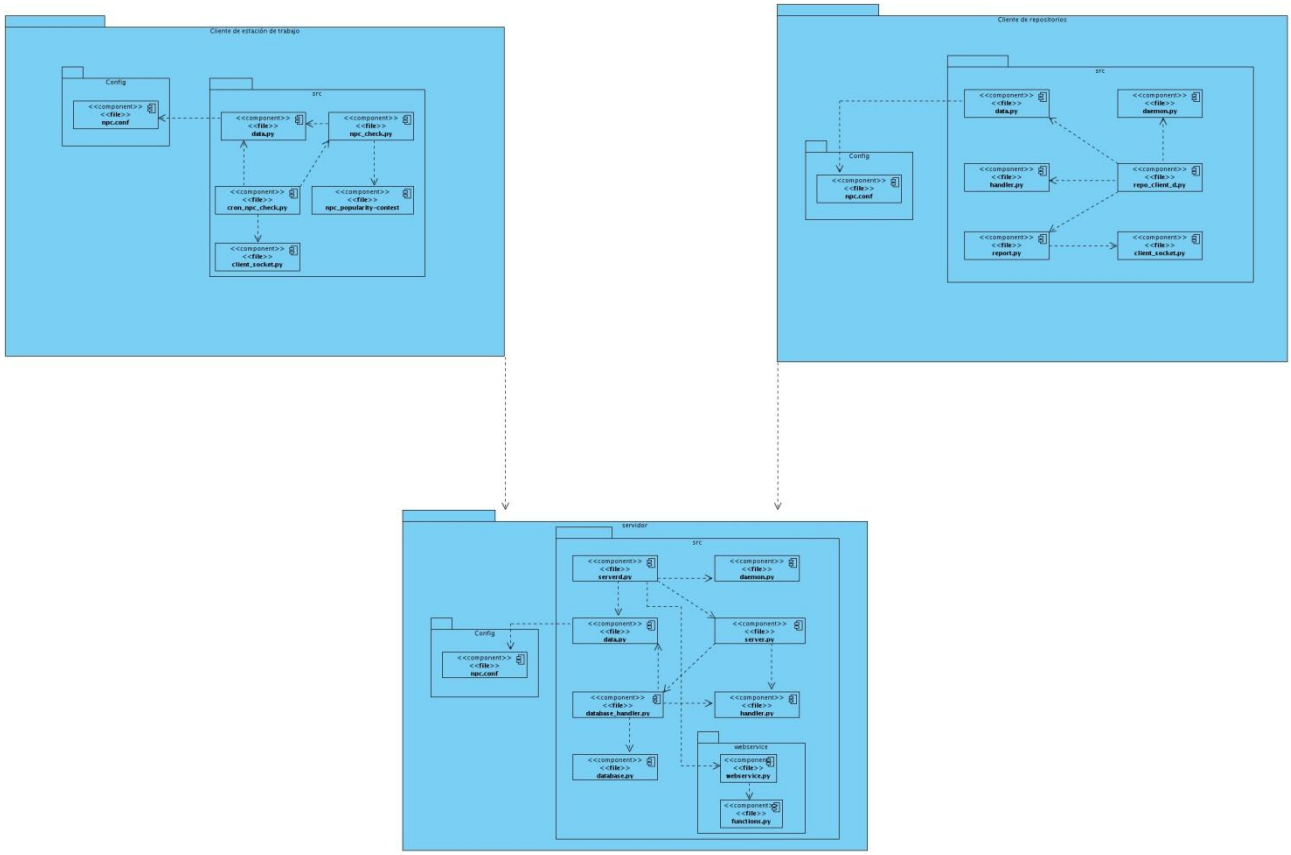
SAMPIERI, Roberto Hernández, COLLADO, Carlos Fernández and LUCIO, Pilar Batista. *Metodología de la Investigación*. IV. S.l.: McGraw-Hill/Interamericana, 2006.

GIFT, Noah and JONES, Jeremy M. *Python for Unix and Linux System Administration*. First Edition. S.l.: O'Reilly Media, [2008].

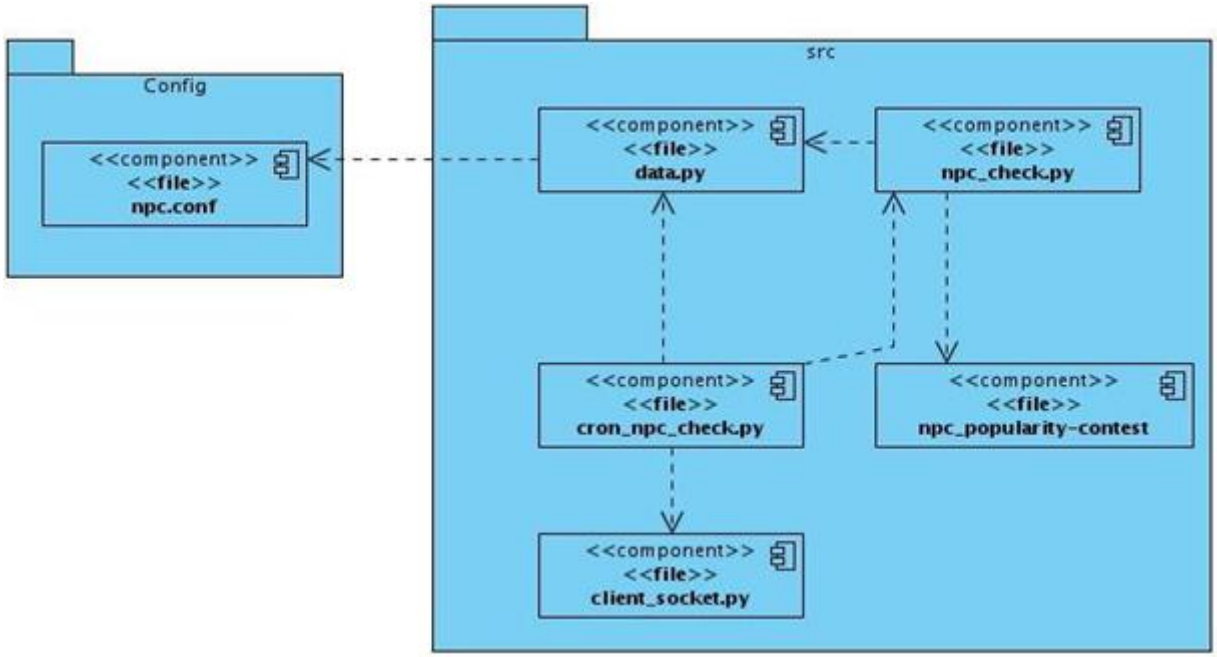
BARRY, Paul. *Head First Python*. First Edition. S.l.: O'Reilly Media, 2010.

Anexos

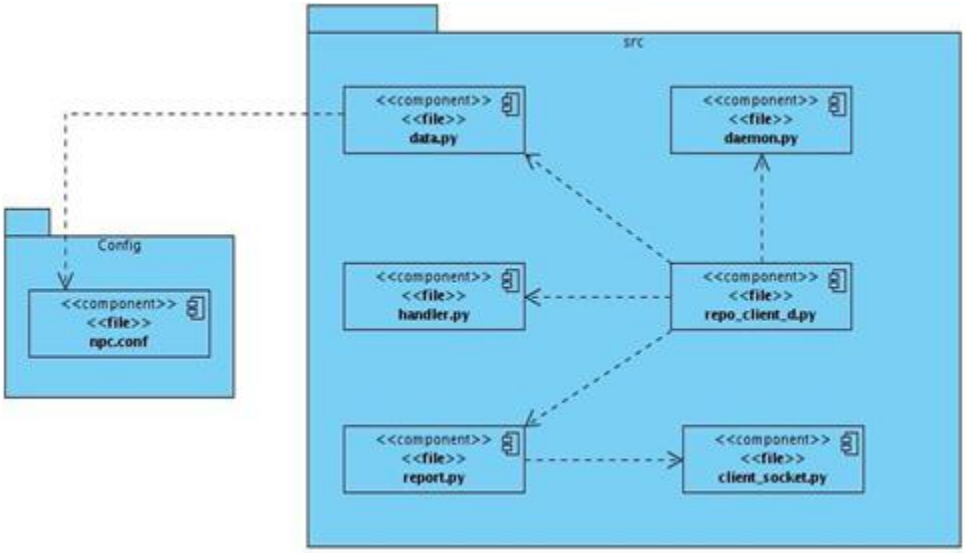
Anexo 1 ***Diagrama de componentes del sistema.***



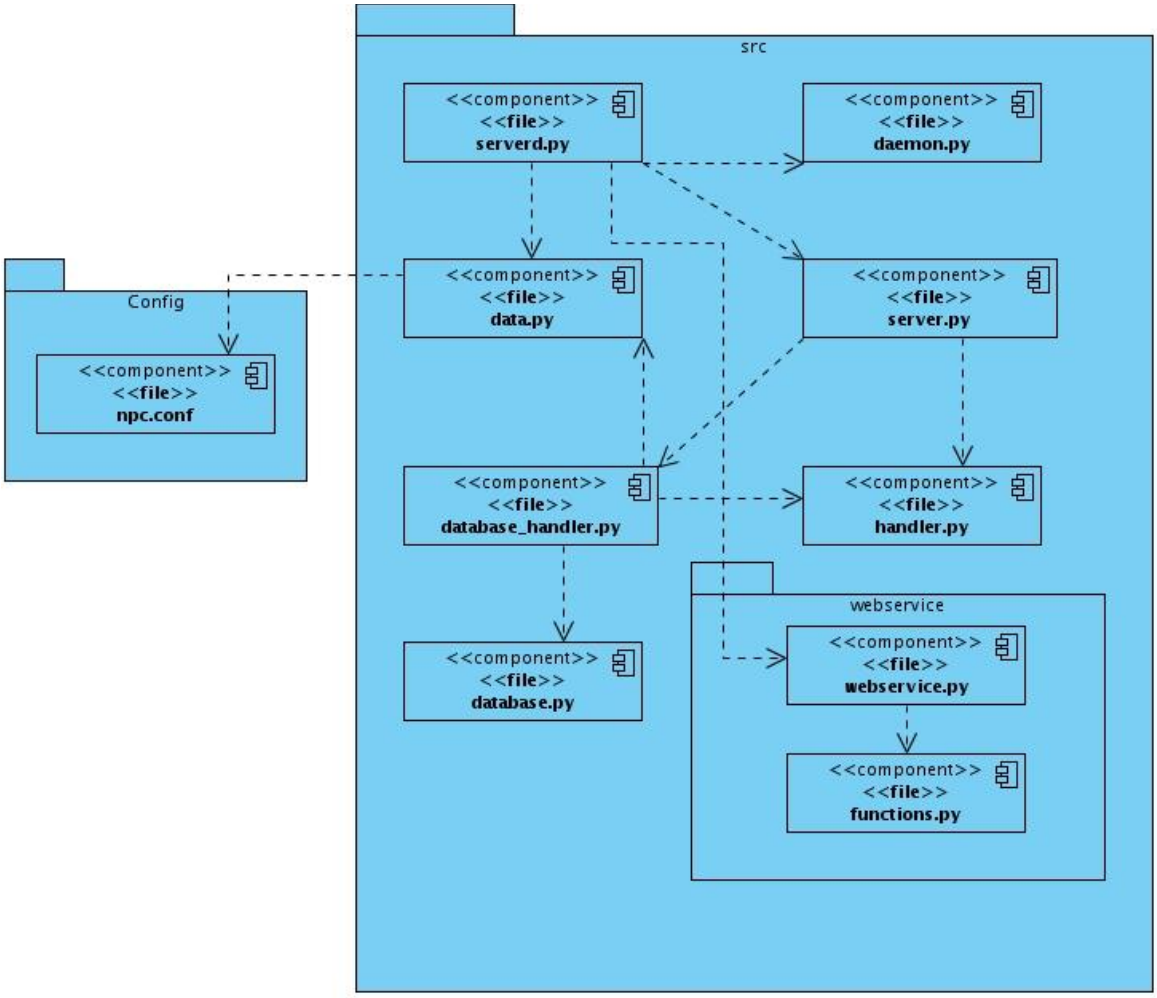
Anexo 2 ***Diagrama de componentes del subsistema Cliente de Estación de Trabajo.***



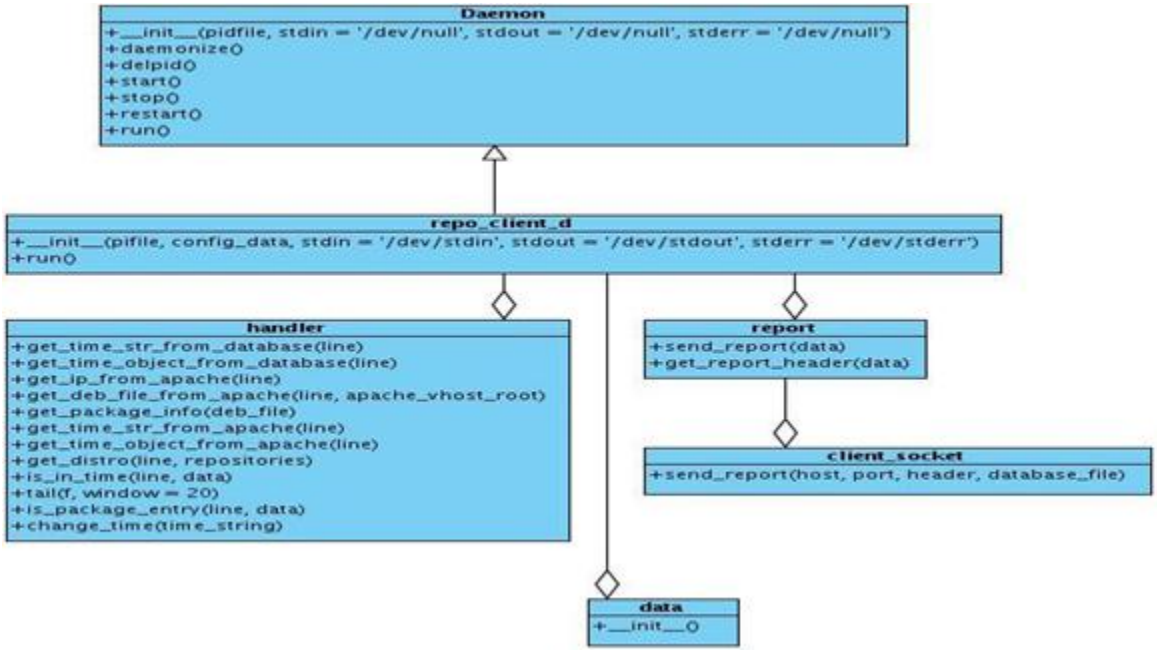
Anexo 3 ***Diagrama de componentes del subsistema Cliente de Repositorios.***



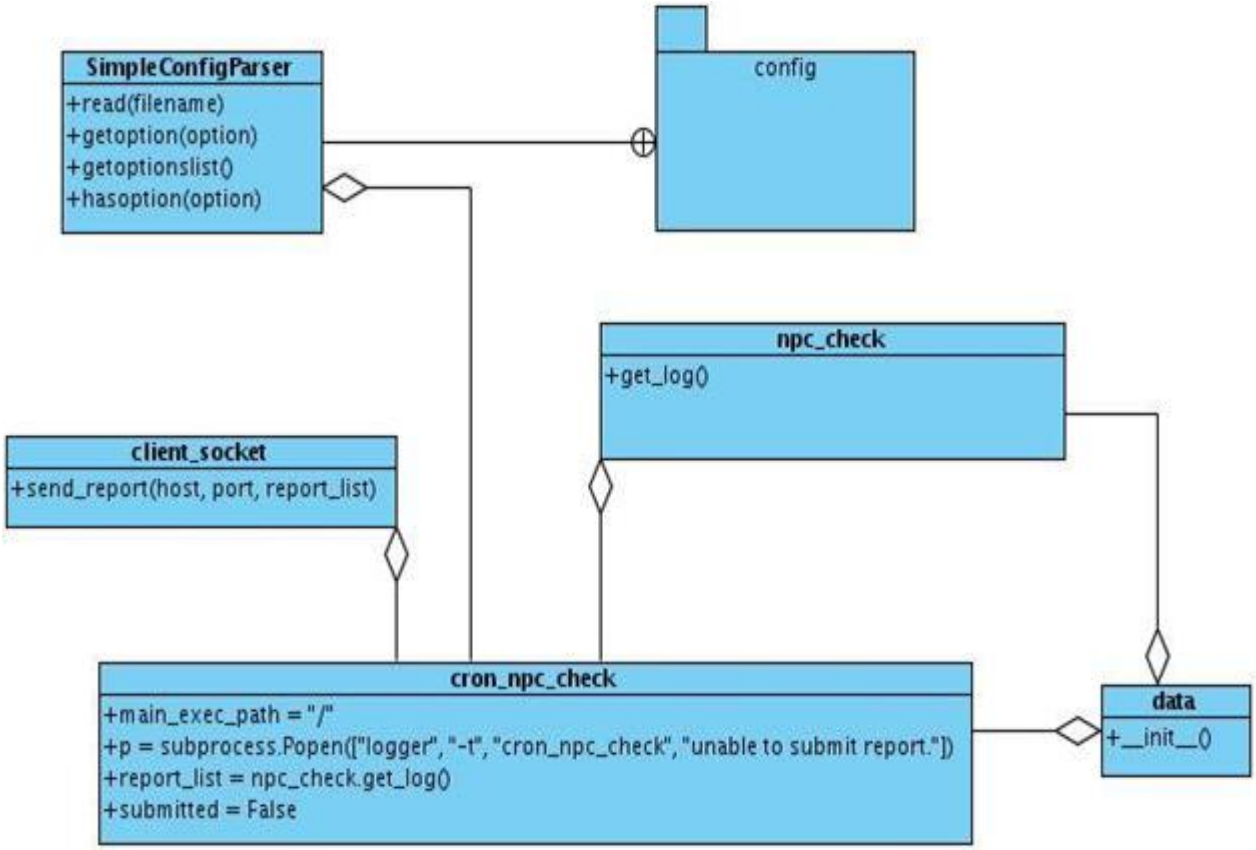
Anexo 4 ***Diagrama de componentes del subsistema servidor.***



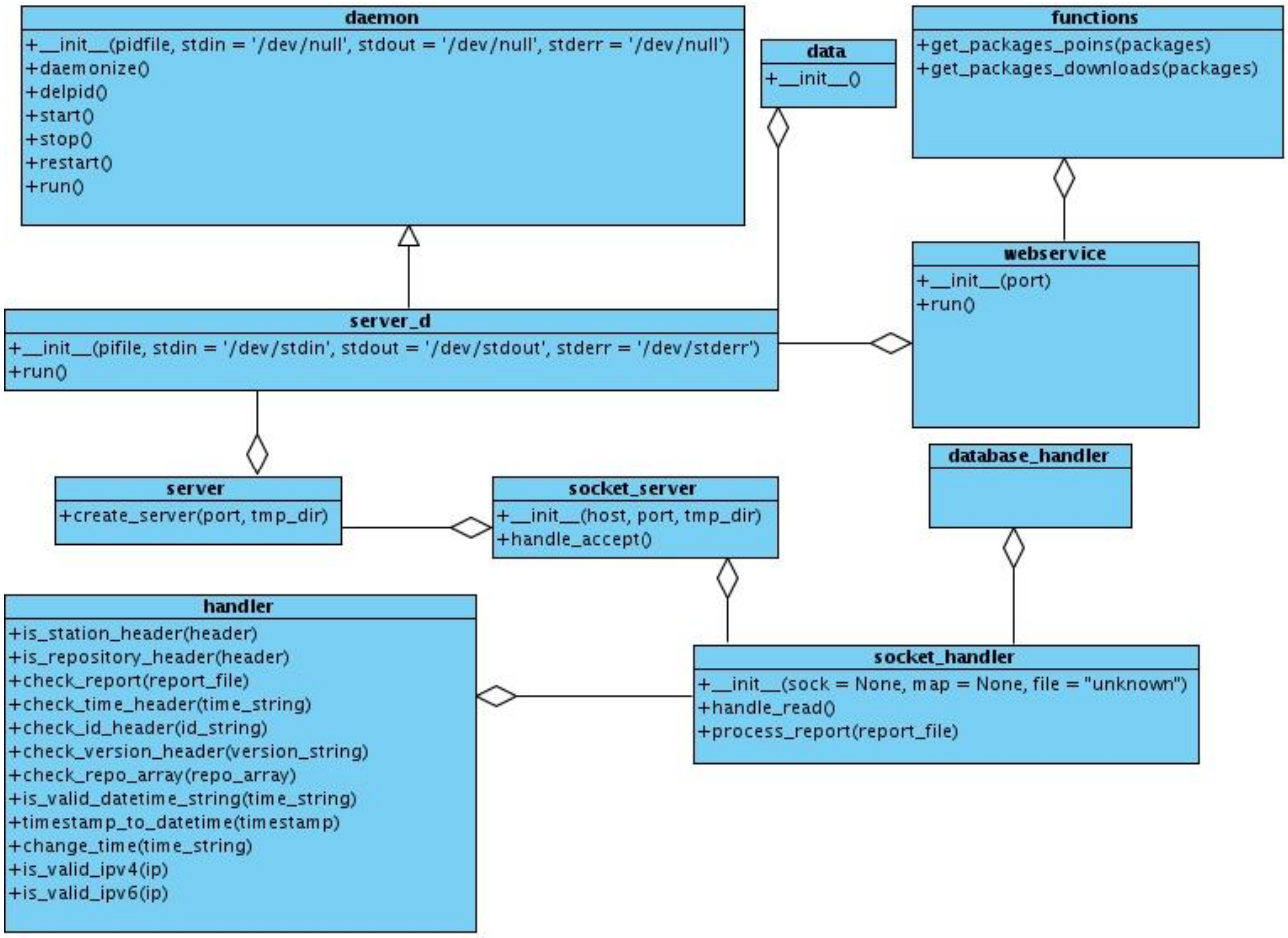
Anexo 5 **Diagrama de clases del diseño del Subsistema Cliente de Repositorio.**



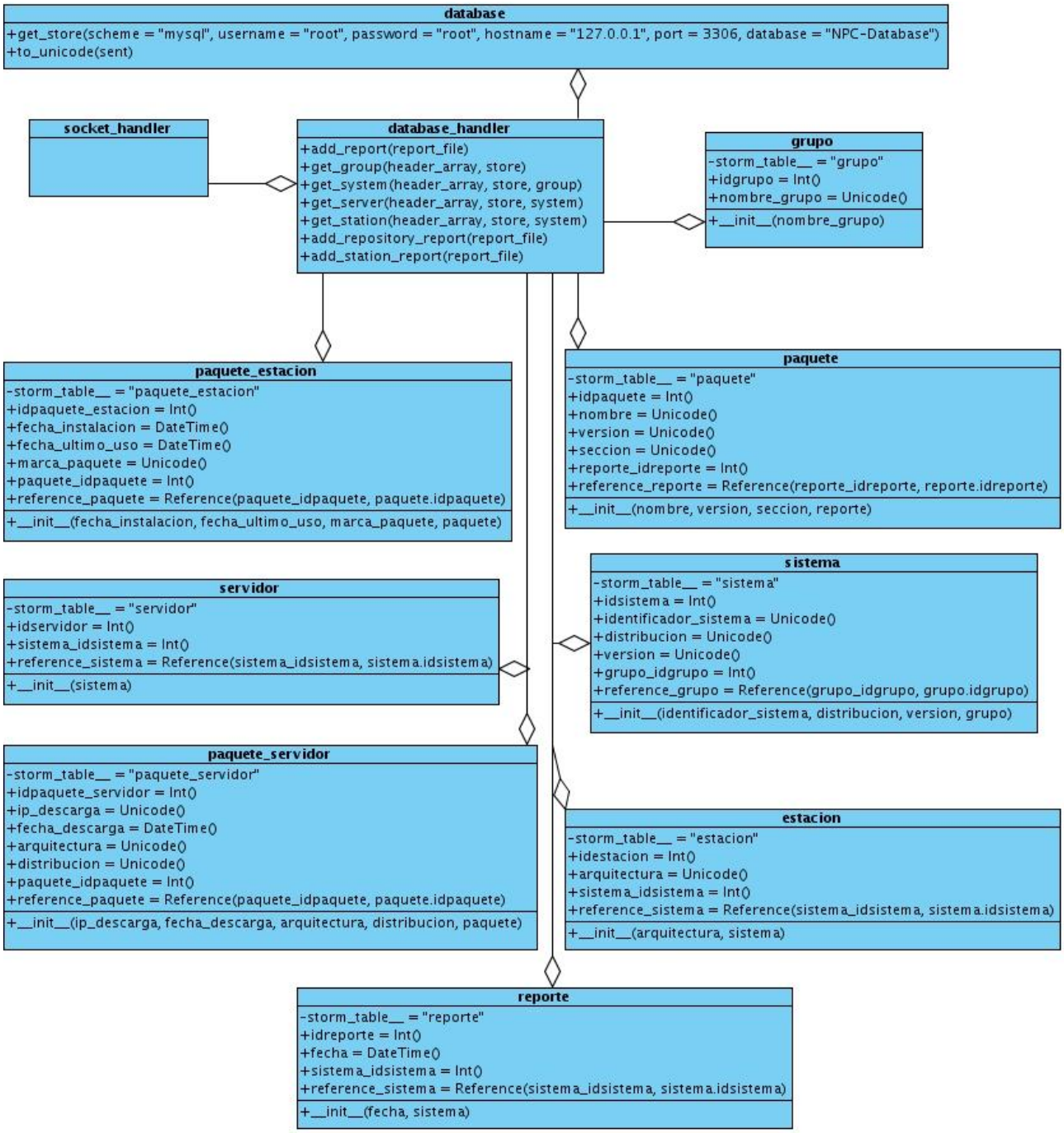
Anexo 6 ***Diagrama de clases del diseño del Subsistema Cliente de estación de trabajo.***



Anexo 7 **Diagrama de clases del diseño del Subsistema Servidor.**



Anexo 7 **Diagrama de clases del diseño del Subsistema Servidor (Continuación).**



Anexo 8 **Diseño de la base de datos.**