

Universidad de las Ciencias Informáticas



“Augeas, propuesta tecnológica para la gestión de archivos de configuración de sistemas GNU/Linux.”

Trabajo de Diploma

Autor(es):

Yuliette Tain Domínguez
Román Miguel Valdivia Genó

Tutor(es):

Ing. Anielkis Herrera Gonzalez
Lic. Dariem Pérez Herrera

Presentada en opción al Título de Ingeniero en Ciencias Informáticas

Ciudad Habana, Cuba
Mayo, 2009

DECLARACIÓN DE AUTORÍA

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

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

Yuliette Tain
Domínguez

Román Miguel
Valdivia Genó

Anielkis Herrera
Gonzalez

Dariem Pérez Herrera

Firma del Autor

Firma del Autor

Firma del Tutor

Firma del Tutor

Agradecimientos

De Yuli:

A mis adorados padres por ser mis guías, mis amigos incondicionales, mi razón de ser y lo más grande que tengo en esta vida.

A mi tata por hacer de las cosas difíciles los momentos más alegres.

A mi mima por quererme tanto y hacer de mi lo que soy hoy.

A mis queridos familiares en especial a mis abuelos que tanto adoro.

A Adis, Baby y Vivi por la incansable batalla para que comiera y por estar en los momentos importantes (que fueron todos), las quiero mucho.

A Román por ser él.

A mis amistades de todo momento, especialmente Harold, el chino, chepa y el feo, por estar siempre.

A Abel, Migue, Miranda, Angel, Micha, Felix y Anielkis, esto también es de ustedes, gracias por enseñarme tanto.

A mi grupo 10504 que son los que más me han enseñado.

A mis tutores por hacer posible este trabajo.

De Román:

A Yuliette por aguantarme estos meses.

A Anielkis y Dariem por sabernos guiar.

A Felix, Abel, Migue, el Micha, Angel, Miranda, el Puro, Baby y Adis por ayudarme a ver el trabajo como una diversión y por haber compartido muchos momentos importantes conmigo.

A la profesora Graciela y a Angel por dedicarnos tiempo cuando lo necesitamos.

A todos y cada uno de los profesores que nos han enseñado a lo largo de estos 17 años de estudio.

A la UCI y a Fidel por permitirnos formar parte de este proyecto futuro.

A todos los que aportaron su granito de arena en ambos trabajos, gracias.

Dedicatoria

De Yuli:

A todas las personas que quiero y me quieren de verdad.

De Román:

A mi papá por haberme convertido en lo que soy.

A mi mamá por todo.

A mi Ama por amarme tanto.

A Karel por ser mi hermano.

Resumen

El sistema Nova requiere del desarrollo de herramientas de configuración propias, durante la implementación de las mismas, los desarrolladores enfrentan grandes dificultades para gestionar los archivos de configuración del sistema. Para darle solución a esta problemática en el siguiente trabajo se propone una herramienta tecnológica llamada Augeas, para estandarizar el trabajo con archivos de configuración del sistema, y así asegurar la calidad de la gestión de estos archivos y facilitar el desarrollo de herramientas de configuración para el sistema operativo Nova.

Índice de contenidos

DECLARACIÓN DE AUTORÍA	2
Agradecimientos.....	3
Agradecimientos.....	3
Dedicatoria.....	4
Dedicatoria.....	4
Resumen.....	5
Resumen.....	5
.....	5
<u>Introducción.....</u>	<u>8</u>
<u>1. Fundamentación del tema.....</u>	<u>11</u>
1.1. La herramienta de configuración.....	11
1.2. Algunas herramientas de configuración del sistema.....	12
1.2.1. Linuxconf	12
1.2.2. YaST.....	13
1.2.3. Webmin.....	14
1.2.4. kadmin.....	15
1.2.5. Gnome-system-tools (GST).....	16
1.2.6. Debconf.....	17
1.3. Módulos y Librerías para el trabajo con los ficheros.....	18
1.3.1. Qt.....	18
1.3.2. Glib.....	19
1.4. Frameworks.....	21
1.4.1. ¿Qué es un framework?	21
1.4.1.1. Ventajas y Desventajas en la utilización de un framework.....	22
1.4.1.2. Ejemplos de frameworks utilizados.....	23
1.5. Estandarización del software.....	25
<u>2. Augeas.....</u>	<u>28</u>
2.1. ¿Qué es Augeas?.....	29
2.2. Arquitectura y Diseño de Augeas.....	30
2.2.1. Estilo arquitectónico en capas.....	30
2.2.1.1. Descripción del diseño por capas.....	31
2.3. Módulos de Augeas: los Lenses.....	34
2.3.1. Programas bidireccionales.....	34
2.3.2. ¿Cómo crear módulos para Augeas?.....	35
2.4. Ideas futuras del proyecto.....	36
<u>3. Augeas en Nova.....</u>	<u>37</u>
3.1. Sistema Operativo Nova.....	37
3.2. Casos de estudio.....	38
3.2.1. Capoeira	38
3.2.1.1. Gestión de ficheros de configuración en Capoeira.....	39
Gestión de smb.conf.....	39
Gestión de fstab.....	40
3.2.2. EcumeniX.....	41
Gestión de hosts.....	43
Gestión de ntp.conf.....	43
<u>Conclusiones.....</u>	<u>44</u>
<u>Recomendaciones.....</u>	<u>45</u>

Referencias Bibliográficas	46
Bibliografía	47
Anexos	50
Anexo 1. Entornos de escritorios más utilizados.....	50
Anexo 2. Listado de usuarios de GNU/Linux por años.....	51
Anexo 3. Listado de compañías que patrocinan a desarrolladores del kernel de Linux.....	52
Anexo 4. Lenguajes de programación de Augeas.....	53
Anexo 5. Arquitectura en capas.....	54
Anexo 6. Capoeira.....	55
Anexo 7. Comparación del método list_share (sin utilizar Augeas) / list_share (utilizando Augeas)	56
Anexo 8. Comparación del método get_fs_data_for_path (sin utilizar Augeas) / get_fs_data_for_path (utilizando Augeas).....	57
Anexo 9. EcumeniX.....	58
Anexo 10. SCR.....	58
Anexo 11. Resultados de la encuesta	59
Anexo 12. Comparación del método transform (sin utilizar Augeas) / transform (utilizando Augeas) utilizado en el fichero hostsparser.py.....	60
Anexo 13. Comparación del método transform (sin utilizar Augeas) / transform (utilizando Augeas) utilizado en el fichero ntparser.py.....	61
Anexo 14. Glosario de Términos.....	62

Introducción

En la Universidad de las Ciencias Informáticas se creó la distribución de GNU/Linux, Nova; con el objetivo de facilitar la migración del país hacia software libre. El proyecto utilizó como distribución base Gentoo, pues brinda una alta capacidad para optimizar y personalizar el sistema, además de garantizar flexibilidad y portabilidad para todas las arquitecturas soportadas por el GCC (*GNU Compile Colection*), entre otros aspectos importantes. A partir de la distribución base se creó y modificó un conjunto de aplicaciones necesarias para garantizar una migración cómoda.

La distribución Nova utiliza para la configuración del sistema las HCSO que brinda Gnome (entorno de escritorio por defecto), o las herramientas del entorno de escritorio que seleccione el usuario. El cambio de entorno de escritorio trae consigo variación en la forma de configurar el sistema, haciendo el trabajo de configuración más complejo para los usuarios.

Para erradicar el problema la distribución Nova tiene entre sus metas suplir la necesidad de herramientas de configuración del sistema propias, que estandaricen la configuración del sistema operativo. El desarrollo de dichas herramientas es un trabajo difícil, pues las aplicaciones de configuración del sistema mantienen un fuerte vínculo con los archivos de configuración.

La situación planteada se vuelve compleja cuando se tienen en cuenta las dificultades que enfrenta cualquier desarrollador a la hora de escribir un código que interactúe con archivos de configuración del sistema GNU/Linux, producto de la variabilidad y diversidad existente en su sintaxis.

A partir de ésta situación problemática surge el siguiente **problema científico**: ¿Cómo facilitar la gestión de los archivos de configuración del sistema Nova?

Como **objeto de estudio** se tiene: las herramientas de configuración del sistema GNU/Linux cuyo **campo de acción** se centra en las herramientas utilizadas para la gestión de archivos de configuración del sistema GNU/Linux y como **objetivo general**, proponer una herramienta que estandarice y facilite el trabajo con los archivos de configuración del sistema Nova.

Esta investigación tiene los **objetivos específicos** siguientes:

- Estudiar el estado actual de las herramientas de configuración y gestión de archivos de configuración de sistemas GNU/Linux.
- Seleccionar una herramienta que permita estandarizar la gestión de archivos de configuración

del sistema Nova.

- Demostrar la necesidad de estandarizar la gestión de archivos de configuración del sistema, en el desarrollo de las herramientas de configuración del sistema Nova.

La **idea a defender** con esta investigación es: La adopción de una herramienta que permita la estandarización de la gestión de archivos de configuración del sistema Nova, permitirá agilizar el desarrollo de las herramientas de configuración de dicho sistema.

Para alcanzar esos objetivos, se han definido las **tareas de investigación** siguientes:

- Sistematización teórica de los antecedentes y estado actual de herramientas que gestionan archivos de configuración del sistema.
- Realización del caso de estudio seleccionado.
- Demostrar con el caso de estudio realizado, la necesidad de la herramienta seleccionada para estandarizar el desarrollo de las aplicaciones de configuración en el sistema Nova.

Los **métodos investigativos** utilizados en el transcurso de la investigación son los siguientes:

- Analítico-Sintético: este método permitió descomponer el problema en partes y luego en subconjuntos para llegar a generalizaciones. Se utilizó para el estudio de la estructura y los componentes de Augeas.
- Análisis bibliográfico: esta técnica permitió realizar una revisión y análisis de las diferentes bibliografías referentes a las herramientas de configuración de sistemas GNU/Linux, así como las utilizadas para la gestión de archivos de configuración; para posteriormente extraer y sintetizar los elementos más significativos teniendo en cuenta el objeto de estudio. Se organizó la bibliografía consultada según la norma ISO 690-Referencia Numérica.
- Histórico-lógico: este método posibilitó conocer los antecedentes y tendencias más actuales de las tecnologías utilizadas para la gestión de los archivos configuración de los sistemas GNU/Linux en el mundo, en Cuba y más específicamente en la UCI. Mediante este método se analizó la trayectoria concreta de la herramienta seleccionada.
- Encuesta: para la realización de esta técnica se utilizó un muestreo probabilístico aleatorio. Con los resultados obtenidos, se creó la base para la investigación de las tecnologías utilizadas en la gestión de ficheros de configuración del sistema GNU/Linux.
- Observación: esta técnica permitió tener un registro visual del comportamiento de la

herramienta en distintos entornos.

Para organizar el presente documento se ha dividido en tres capítulos estructurados como se muestra a continuación:

Capítulo 1. Fundamentación teórica. En este capítulo se realiza un estudio de las tecnologías utilizadas en la actualidad para trabajar con los archivos del sistema GNU/Linux.

Capítulo 2. Augeas. Este capítulo se centra en la investigación del framework para la gestión de archivos de configuración del sistema, Augeas.

Capítulo 3. Augeas en Nova. En este capítulo se hace una demostración del aporte que brinda Augeas a Nova. Se explican las ventajas de utilizar Augeas en el desarrollo de Capoeira y de Ecumenix, que son las aplicaciones de configuración del sistema Nova seleccionadas como caso de estudio.

Este documento contiene además, **Conclusiones, Recomendaciones, Bibliografía, Anexos y Glosario de Términos.**

En el documento se utilizaron las siguientes convenciones:

SO: Sistema Operativo.

HCSO: Herramientas de configuración del sistema operativo.

API: Interfaz para programación de aplicaciones.

UCI: Universidad de las Ciencias Informáticas.

1.Fundamentación del tema

Entre las principales características de los sistemas basados en Unix está la alta capacidad de configuración que brindan; tomando como archivo además de los datos a los directorios e incluso a los componentes de la computadora. Imponiendo que para configurar cualquier propiedad del sistema operativo sea necesario el trabajo directo con archivos de configuración. Como la gestión manual de archivos tiene grandes complejidades para los usuarios, la configuración del sistema se torna difícil sin el uso de herramientas de configuración.

Algunas distribuciones de GNU/Linux han desarrollado de forma independiente, herramientas para facilitarle a los usuarios la interacción con particularidades de su SO. Como consecuencia en la actualidad, existen variadas herramientas de configuración; estas difieren en la tecnología que aplican, los lenguajes en que están programadas y su usabilidad.

Aunque existan varias tecnologías para la gestión de los ficheros requeridos por las aplicaciones de configuración del sistema como:

- Librerías
- Módulos
- Lenguajes de programación con funcionalidades para la gestión de ficheros.

El desarrollo de dichas aplicaciones no es un trabajo sencillo para los programadores.

En el presente capítulo se describe y se hace un estudio de las herramientas más utilizadas para la configuración de los diferentes SO de GNU/Linux, así como las tecnologías utilizadas por los desarrolladores para el manejo de los ficheros que controlan las propiedades de dichos sistemas.

1.1.La herramienta de configuración.

Para mejor comprensión de la investigación, partiendo del estudio realizado sobre las herramientas utilizadas para la configuración de sistema, y las funciones que brindan las mismas; los autores de la investigación definen estos artefactos como: *“Software utilizado para manejar las propiedades del*

sistema.”

Con esta definición queda explícito cual es el principal objetivo de la herramienta de configuración del sistema.

“Si utiliza alguna de estas herramientas notará que el control no es de nadie si no suyo, y enseguida descubrirá que configurar y administrar su Linux es fácil... Definitivamente, pocos son los sistemas operativos que conjugan bien flexibilidad casi total, con verdadera comodidad para el usuario.”(2) Los autores de esta cita, hacen referencia a las facilidades que brinda el trabajo con las herramientas de configuración. Afirmando la carencia en la actualidad de sistemas que tengan la capacidad de abarcar ambas características: la flexibilidad para modificar el sistema y la facilidad para realizar los cambios.

“El concepto en torno al cual gravita la usabilidad es la calidad de uso. No se trata de "pelearse" con el ordenador para conseguir que haga lo que uno quiere sino todo lo contrario...” (3).

El profesor Xavier explica la importancia que tiene para un software brindarle facilidades de uso al usuario. Es decir que el software esté en función del usuario y no al revés.

Brindarle usabilidad a las distribuciones que las utilizan es otro de los objetivos de las herramientas de configuración del sistema. A continuación se describen las herramientas más utilizadas para la configuración, en sistemas GNU/Linux.

1.2. Algunas herramientas de configuración del sistema.

La configuración del sistema GNU/Linux puede realizarse de diferentes formas:

- Mediante terminales.
- Mediante aplicaciones gráficas
- Mediante aplicaciones web

Siendo las dos últimas las más usadas por los económicos, los médicos, los profesores, en fin los usuarios comunes. Entre las herramientas de configuración del sistema GNU/Linux más utilizadas se encuentran:

1.2.1. Linuxconf

Linuxconf que se puede utilizar para administrar el sistema, permitiendo añadir y manipular cuentas de usuarios, visualizar actividades del sistema, controlar el arranque del sistema, entre muchas otras

cosas. Dentro de las propiedades que presenta, están las facilidades de accionamiento:

“Las distintas opciones para acceder a la herramienta son:

- Por línea de comando, lo cual facilita la manipulación de la configuración del sistema.
- Sistemas de ventanas X, que posee interfaz gráfica.
- Vía Web, que permite la administración remota a través de navegación web.”(4)

La herramienta está “escrita en C++, pero otros lenguajes pueden ser usados para escribir módulos vinculados, como por ejemplo: Python, Perl o Bash. Linuxconf divide los archivos de configuración en subsistemas, y administra el sistema por módulos separados.” (4)

Se distribuye bajo licencia libre, GPL (GNU General Public License), utilizando LGPL (GNU Lesser General Public License) para los módulos vinculados, es decir los componentes vinculados a Linuxconf dinámicamente en tiempo de ejecución. Esos componentes tienen pleno acceso a la API de Linuxconf. Para el manejo de la configuración del sistema tiene tres módulos principales: misc, dialog y userconf.

- Dialog: módulo encargado de la interfaz de Linuxconf.
- Userconf: se encarga de la configuración de los usuarios.
- Misc: contiene un conjunto de librerías con distintas funcionalidades, una de estas es Config File, que brinda las funcionalidades para el trabajo con los ficheros de configuración.

Linuxconf es de las herramientas más utilizadas aunque también existen otras, como YaST, de la distribución SuSe, que facilita la administración del sistema y la instalación de software.

1.2.2. YaST

YaST provee una interfaz desarrollada en Qt, conveniente para la administración de software, hardware, redes, configuración de sistema, seguridad, cuentas de usuario, entre otras acciones de configuración; incluso es la herramienta que conduce al usuario durante la instalación de SuSe. “Las formas de accionamiento son por línea de comando y por interfaz gráfica (sistema de ventanas X).”(5)

En sus inicios fue distribuida bajo licencia privativa, pero en la actualidad se liberó bajo la licencia GPL, de GNU. “Está desarrollada mayormente en C++ y YCP (YaST Control Protocol, es el lenguaje de programación de YaST para el desarrollo de la interfaz de usuario y la lógica), aunque se puede

utilizar Perl para el desarrollo de módulos y en menor medida Python y Ruby.” (5)

Esta herramienta realiza la gestión de los archivos de configuración a través de SRC (Configuración del sistema de repositorios). Entre las principales características de esta capa de abstracción están: “

- Es un sistema de captación de datos
- Traduce los datos desde / a un formato común (YCP)
- Brinda una interfaz para todo el sistema
- Los agentes (que son pequeñas aplicaciones) pueden ser implementados en diferentes lenguajes (C + +, Perl, Bash, entre otros)
- Utiliza comandos simples - Leer, Escribir, Ejecutar.” (6)

La investigación del funcionamiento de la herramienta YaST con SRC arrojó los siguientes resultados: YaST contiene un framework base que maneja las aplicaciones realizadas para cada funcionalidad del sistema. Con el SCR (que son pequeños programas escritos en Perl, Bash, o C, que realizan funciones específicas con los ficheros, y son unidos por Ini_Agent o Any_Agent) maneja la gestión de los ficheros de configuración del sistema. (Ver [Anexo 10](#))

1.2.3. Webmin

Otra de las principales herramientas de configuración del sistema es *Webmin* que “es un panel de control vía web escrito en Perl, para administrar sistemas operativos. Este panel de control viene a simplificar tareas que normalmente se realizan a través de la consola del sistema. Con él se pueden configurar aspectos internos de muchos sistemas operativos, como los usuarios, las cuotas de espacio, servicios, archivos de configuración, apagado del equipo, etcétera. Así como modificar y controlar muchas aplicaciones de código abierto, como el servidor web Apache o los servidores DNS, Samba, DHCP, entre otros.

Está librado bajo licencia BSD (Berkeley Software Distribution) y construido a partir de módulos, los cuales tienen una interfaz a los archivos de configuración y el servidor Webmin. Esto hace fácil la adición de nuevas funcionalidades sin mucho esfuerzo. Debido al diseño modular de Webmin, es posible escribir extensiones para configuración de escritorio.” (7)

El resto de las herramientas descritas también están construidas a partir de módulos independientes, que se encargan de las distintas funciones y de la actualización de los archivos de configuración.

Para la confección de estos módulos y librerías son utilizadas tecnologías independientes que existen para el manejo de los ficheros de configuración.

Además del YaST, el Linuxconf y el Webmin, están las herramientas de administración del sistema que han desarrollado algunos entornos de escritorio. Cada distribución de GNU/Linux selecciona él o los entornos de escritorio que va a utilizar; en dependencia de sus necesidades y las preferencias de los usuarios.

En la actualidad existen muchos entornos de escritorio libres, según los resultados alcanzados, en una encuesta realizada por la distribución SuSe (Ver [Anexo1](#)), estos son los entornos de escritorios más utilizados:

KDE: *“Es un entorno de escritorio, una colección de programas, tecnologías y documentación que intentan facilitar el uso de los ordenadores a sus usuarios. KDE está pensado para estaciones de trabajo UNIX y se caracteriza por la transparencia en la comunicación por redes y una filosofía moderna de trabajo.”* (8)

Gnome: *“Es un entorno de escritorio amigable que permite a los usuarios una configuración fácil de sus sistemas. GNOME incluye un panel (para iniciar las aplicaciones y mostrar su estado), un escritorio (donde pueden colocarse los datos y las aplicaciones), un conjunto de herramientas y aplicaciones de escritorio estándar y un conjunto de convenciones para facilitar la creación de aplicaciones que sean consistentes y colaboren unas con otras.”* (8)

XFCE. Es un entorno de escritorio ligero para sistemas operativos Unix. Su objetivo es ser rápido y ligero, sin dejar de ser visualmente atractivo y fácil de usar. Se basa en el kit de herramientas GTK+ al igual que GNOME.

Los entornos de escritorio KDE y Gnome poseen algunas herramientas de configuración propias del sistema. A continuación se describen para complementar la investigación realizada a las herramientas de configuración más utilizadas del sistema GNU/Linux.

1.2.4. kdeadmin

“Este meta paquete incluye una colección de herramientas de administración del sistema que se proporcionan en la publicación oficial de KDE.”(9)

Kdeadmin es un conjunto de aplicaciones para configuración, que utiliza el entorno de escritorio KDE. Las aplicaciones son:

Kpackage: es un gestor de paquetes.

Kcron: es un planificador de tareas.

Kuser: es un gestor gráfico de usuarios.

Kwuftpd: es un editor de FTPD.

Ksysv: es un editor de Sys V-Init (10)

Las aplicaciones están desarrolladas en C++ con Qt, al igual que KDE. Y son utilizadas por distribuciones como Ubuntu, Debian, Fedora, entre otras.

1.2.5. Gnome-system-tools (GST)

“Anteriormente conocido como el programa de instalación de Ximian, la GST es un conjunto integrado de herramientas destinadas a facilitar el trabajo, que constituye el equipo de administración en un sistema UNIX o Linux. Está pensado para ayudar a los nuevos usuarios de Linux o UNIX, o a los administradores del sistema. Las herramientas del sistema de GNOME son libres, distribuidas bajo los términos de la licencia GNU General Public License.” (11)

El paquete de administración contiene un grupo de aplicaciones que se utilizan para (11):

- Administrar los usuarios o grupos de sistema.
- Configurar la fecha y el tiempo.
- Administrar las redes.
- Administrar las carpetas compartidas a través Samba o NFS.
- Administrar los niveles de ejecución.

Gnome-system-tools brinda usabilidad al sistema con el que se esté trabajando, las herramientas que contiene son muy utilizadas por los usuarios de GNU/Linux, pues además de brindar flexibilidad para la configuración son muy sencillas de manipular.

Otra tecnología informática que se utiliza para la configuración de los archivos del sistema es el paquete Debconf.

1.2.6. Debconf

Debconf es un gestor de configuración para paquetes de la distribución Debian GNU/Linux, y está integrada con el sistema de gestión de paquetes, dpkg.

Durante la instalación del paquete, debconf hace preguntas al usuario usando las repuestas para configurar de forma automática el nuevo paquete. El nivel de las preguntas hechas depende en cómo está configurado debconf. Generalmente hay tres niveles de preguntas disponibles:

- High, hace solamente las preguntas cruciales para que funcione la configuración.
- Medium, realiza las preguntas cruciales, más las preguntas necesarias para las personalizaciones más comunes.
- Low, hace preguntas para todos los detalles.

Siempre es posible cambiar la configuración de debconf con el comando `dpkg-reconfigure debconf`. Esta es una herramienta muy utilizada para la configuración de los paquetes instalados en Debian.

Este editor no brinda soporte para Entropy, que es el gestor de paquetes que utiliza el sistema Nova. Por lo cual no es una opción para darle solución a la problemática planteada.

Las herramientas descritas, son las más utilizadas para la configuración del sistema. La gestión de los ficheros de estas herramientas está basada de modo general, en librerías y módulos creados para facilitar ese trabajo.

Con el objetivo de comprobar cuáles son las herramientas más utilizadas para la gestión de ficheros que interactúan con aplicaciones, por los desarrolladores de software libre de la UCI, los autores de la investigación realizaron una encuesta. Para la cual se seleccionó una muestra aleatoria de 2 proyectos de la facultad de software libre de la UCI, de los cuales fueron encuestados 10 desarrolladores por proyecto para un total de 20 desarrolladores.

Teniendo en cuenta lo expuesto anteriormente las encuestas arrojaron los siguientes resultados: el 100% de los encuestados habían gestionado archivos durante el desarrollo de alguna aplicación. El 75% afirma realizar la gestión de ficheros durante el desarrollo de aplicaciones, con las librerías y las funcionalidades que brinda el lenguaje de programación que utilice; el 5% utiliza módulos diseñados para el manejo de ficheros y los restantes utilizan ambas tecnologías.

Con los resultados se comprobó que los desarrolladores utilizan generalmente las librerías que brindan los lenguajes de programación utilizados. (Ver [Anexo 11.](#))

A continuación se describen algunas de las librerías utilizadas por los desarrolladores de software, para la gestión de los archivos de configuración; en sistemas operativos GNU/Linux.

1.3. Módulos y Librerías para el trabajo con los ficheros.

En la actualidad la mayoría de los lenguajes contienen módulos y librerías para el manejo de ficheros, como por ejemplo, Perl tiene una variedad de módulos útiles para el manejo de ficheros, entre los cuales se pueden citar “*File que determina el tipo de fichero, File::Comments que reconoce los formatos de archivo y el formato específico de los comentarios extraídos, File::Monitor que proporciona una interfaz simple para el seguimiento de uno o más archivos de información y los cambios que se hacen a ellos y File::Path que crea o elimina estructuras de directorios.*” (12)

Java también brinda muchas posibilidades para el trabajo con los ficheros, como por ejemplo el paquete *java.io* contiene las clases necesarias para la comunicación del programa con el exterior. La entrada y la salida de datos del programa se manejan con clases derivadas de *Reader* y *Writer*. “*Para archivos de texto son preferibles FileReader (desciende de Reader) y FileWriter (desciende de Writer). La clase File también contiene diversas funciones para obtener información de un fichero existente, entre otras.*” (13)

C++ brinda fundamentalmente dos clases para la entrada y salida de ficheros, correspondientes a tres tipos de flujos. Estas son *ofstream* para la salida e *ifstream* para la entrada.

Los lenguajes de programación mencionados son de los más utilizados para el trabajo con ficheros, aunque la mayoría de los lenguajes de programación existentes, tienen propiedades para esto. Entre las tecnologías utilizadas por los programadores para la gestión de archivos se encuentran las librerías:

- Qt
- Glib

1.3.1. Qt

“Qt es una biblioteca multiplataforma, creada para desarrollar interfaces gráficas de usuarios. Fue creada por la compañía noruega Trolltech (una compañía de software fundada en el año 1994). Es

desarrollada en el lenguaje C++ de forma nativa, pero existen módulos para otros lenguajes de programación como C, Python (pyqt), Java (Qt Jambi), Perl (PerlQt), entre otros. Es una librería totalmente Orientada a Objetos.” (14)

Desde junio de 2008 es propiedad de Nokia. Qt es utilizada en el desarrollo de muchas aplicaciones, como por ejemplo el entorno de escritorio KDE o el reproductor de multimedia VLC.

También brinda numerosas funcionalidades a los desarrolladores, para la implementación de aplicaciones gráficas. Entre ellas la clase `QFileInfo` que “*proporciona acerca de un archivo:*

- el nombre.
- *el cargo (ruta) en el sistema de archivos.*
- *sus derechos de acceso.*
- *si se trata de un directorio, un enlace simbólico, etcétera.*
- *el tamaño del archivo.*
- *la última modificación.*
- las veces que ha sido leído.” (15)

1.3.2. Glib

Existen otras librerías como Glib, que “*proporciona los bloques básicos para construir aplicaciones y bibliotecas escritas en C. Proporciona el sistema de objetos básico usado en GNOME, la implementación del bucle principal, y un gran conjunto de funciones de utilidad para cadenas y estructuras de datos comunes.*” (16)

Uno de los mayores beneficios de usar Glib es que provee una interfaz de plataforma independiente que permite que el código pueda ser usado en diferentes sistemas operativos. Otro aspecto de Glib es la amplia gama de tipo de datos que deja disponible al desarrollador.

Glib además de proporcionar varios tipos de datos, también dispone de numerosos tipos de funciones, entre las que se encuentran funciones de gestión de archivos, soporte de internacionalización, cadenas de caracteres, advertencias, banderas de depuración y carga dinámica

de módulos, sólo por nombrar algunas.

Las herramientas descritas son muy importantes para el desarrollo de aplicaciones que necesiten gestionar ficheros. Pero aún con las facilidades que brindan, la aplicación que interactúa con ficheros de configuración sigue siendo vulnerable ante cambios en la estructura del fichero gestionado, producto a la alta cohesión que existe entre las herramientas y los ficheros.

Para eliminar esta vulnerabilidad se necesita una herramienta que brinde mayores facilidades a los desarrolladores para la gestión de los ficheros del sistema en GNU/Linux. Aunque algunas de las HCSO que se describieron fueron desarrolladas con la utilización de librerías y módulos para la gestión de ficheros; en muchas ocasiones su desarrollo estuvo basado en algunos frameworks.

Con el objetivo de estudiar las opciones existentes para facilitar el trabajo con los ficheros de configuración, se investigó sobre algunos frameworks.

1.4. Frameworks

1.4.1. ¿Qué es un framework?

“En general, un framework es una estructura real o conceptual que pretende servir como soporte o guía para la construcción de algo que expande su estructura en algo usable.” (2) Los frameworks al ser estructuras de soporte se pueden utilizar en todas las esferas. En la informática existen frameworks para distintas ramas, como por ejemplo el FAIA, que es un framework para la enseñanza de agentes de inteligencia artificial, o CAPEC que es un framework sobre patrones de ataques informáticos.

Para el desarrollo de software se han creado distintas definiciones de framework. El arquitecto Otálora define los framework como *"una técnica de reutilización que busca potenciarla a gran escala, ayudando a los desarrolladores en la creación y mantenimiento de soluciones a problemas dentro de un dominio."* Aunque existen frameworks que pueden abarcar más de un dominio, este concepto explica una de las características fundamentales que brindan los mismos, la reutilización de código fuente.

En el artículo *"El Desarrollo del Framework Orientado al Objeto"* los autores plantean: *"El desarrollo del framework está ganando rápidamente la aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente (source code)."* (17) La reutilización en la actualidad es de gran utilidad, pues además de ser uno de los grandes avances de la programación, brinda la posibilidad a los desarrolladores de enfocarse en los requerimientos del sistema que se está desarrollando.

Otra de las características que identifican a estos marcos de apoyo es que posibilitan un diseño abstracto a seguir para el software que se va a desarrollar según define el siguiente concepto planteado en el mismo artículo:

"El framework captura las decisiones de diseño comunes en un tipo de aplicaciones, estableciendo un modelo común a todas ellas, asignando responsabilidades y estableciendo colaboraciones entre las clases que forman el modelo." (17)

Se considera que los frameworks además del diseño común brindan la posibilidad de que las aplicaciones que se creen a partir de su utilización sean aplicaciones estándares entre ellas, esto

proporciona calidad y estabilidad a las aplicaciones desarrolladas.

Cuando se va a desarrollar una aplicación surge la duda de si es necesario utilizar un framework, o no. Lo cual es cuestionable en dependencia de la complejidad de la implementación, y el ámbito para el cual se está trabajando.

Un desarrollador puede crear toda una aplicación sin ningún framework, quizás sea tan sencilla que no lo considere necesario. Sin embargo a medida que crece la aplicación este seguirá determinadas pautas que faciliten el trabajo, como: separar la capa de presentación de la capa lógica, mantener una sintaxis coherente, etcétera. Por tanto la evolución natural será que de algún modo se construya su propio framework.

Para definir los pros y los contras del desarrollo del framework, los autores llevaron a cabo una comparación entre las desventajas y las ventajas que puede acarrear para un proceso de desarrollo de software la utilización de un framework en la implementación de aplicaciones.

1.4.1.1. Ventajas y Desventajas en la utilización de un framework

Entre los distintos criterios revisados, la mayoría coinciden con que las principales desventajas que trae usar un framework en el desarrollo de una aplicación son:

- La dependencia del código fuente de una aplicación con respecto al framework. Si se desea cambiar de framework, la mayor parte del código debe reescribirse.
- La demanda de grandes cantidades de recursos computacionales dada la característica de reutilización de los frameworks que tiende a generalizar la funcionalidad de los componentes. El resultado es la inclusión de características que están "de más", provocando una sobrecarga de recursos que se hace más grande mientras más amplio es el campo de reutilización.

Aunque en algunos casos, utilizar un framework conlleva a estas consecuencias, hay que tener en cuenta las ventajas y las facilidades que implica basar el desarrollo sobre un framework. Como por ejemplo: (18)

- Modularidad y reducción de la complejidad: La aplicación está formada por subsistemas especializados en distintos aspectos fundamentales de toda aplicación (persistencia, presentación, manejo de logs)
- Fortaleza al cambio: Los módulos pueden ser evolucionados o cambiados conservando la arquitectura global de la aplicación.

- Documentación: La documentación del framework promueve el uso correcto del mismo y disminuye el esfuerzo necesario para el mantenimiento.
- Estructura: El desarrollo basado en frameworks establece una estructura sobre la cual las aplicaciones pueden ser construidas, liberando al desarrollador de tomar el 100% de las decisiones de diseño.
- *Distribución de funciones: Permite el trabajo de desarrollo de forma paralela ya que la solución puede desarrollarse como un conjunto de piezas independientes que encajarán en el framework usado.*
- Eficiencia: El desarrollador puede concentrarse en los requerimientos funcionales de la aplicación.
- Aplicaciones ricas: Posibilidad de dar más funcionalidad a los usuarios de la aplicación.

Tomando en cuenta el criterio anterior, el trabajo con los frameworks proporciona numerosas ventajas, y más aún cuando se necesita estandarizar el desarrollo de las aplicaciones que se van a crear.

La utilización de un framework para el desarrollo de las herramientas que se necesitan para el manejo efectivo de las propiedades de un SO GNU/Linux traería muchas ventajas para los desarrolladores, como por ejemplo: la facilidad de centrarse en los requisitos que se necesitan, que a su vez permite una mayor producción. Además traería ventajas para la calidad de las aplicaciones creadas, pues permitiría la estandarización de su desarrollo, utilizando módulos que pueden ser evolucionados o cambiados sin perjudicar la arquitectura global de la aplicación. Ejemplo de esto se pueden ver las herramientas de administración de KDE, pues su desarrollo está basado en algunos frameworks:

1.4.1.2. Ejemplos de frameworks utilizados.

El proyecto KDE utiliza Solid framework, y le trae numerosas ventajas partiendo de que “es independiente de la plataforma en donde corra (Linux, MS Windows, Unix y Mac OS).” (18) Además de brindarle a sus desarrolladores la ventaja de crear una capa de abstracción para los componentes que utilizan.

Con el uso del framework Solid las aplicaciones no tienen que preocuparse de como acceder al hardware, además de realizar esa operación de manera uniforme sin importar la plataforma que utilice. Algunas de las aplicaciones que lo utilizan son:

- Acceder a dispositivos de audio.
- Acceder a dispositivos removibles: memorias USB, reproductoras de música (genéricos), entre otros.
- Acceder una cámara web.(18)

KDE utiliza también para la multimedia PHONON, que es una capa para otros “motores” de multimedia, este disminuye la cantidad de código que un desarrollador de multimedia tiene que escribir.

Estos son algunos de los que este proyecto utiliza para agilizar y estandarizar el desarrollo de sus aplicaciones.

También existe Augeas; framework desarrollado por Red Hat en el 2007 para el manejo de ficheros de configuración del sistema GNU/Linux.

“Es un framework para la gestión de archivos de configuración. Incluye un API y una herramienta de línea de comando. Aún es de bajo nivel, ya que está obligado a archivos individuales, y muchos objetos OS (como, por ejemplo, una cuenta de usuario) puede requerir cambios en varios archivos y otras acciones (como la creación de un directorio de inicio). Augeas parece una herramienta valiosa.” (19)

La herramienta facilita la gestión de los ficheros de configuración del sistema. Con su utilización se crea una capa de abstracción entre la herramienta y el sistema, que le facilita la implementación de aplicaciones que necesiten gestionar ficheros de configuración al desarrollador.

El desarrollo sobre frameworks es una técnica de reutilización que está tomando auge rápidamente, por las ventajas que trae su utilización. Aunque existen desarrolladores, que prefieren continuar implementando las aplicaciones desde cero, antes de dedicarle tiempo al aprendizaje de un framework.

Para adoptar o desarrollar un framework, es necesario un estudio que verifique la necesidad de estandarizar el software que se va a desarrollar, y el grado de complejidad que tiene la implementación de este software. Así se evita el proyecto el gasto de recursos en caso de que sea innecesario, o la pérdida de tiempo en la implementación en caso de que sí se necesite un framework.

Partiendo del estudio realizado se puede concluir que la utilización de un framework que se ajuste a

las necesidades del proyecto que lo use, trae aumento en la producción y la calidad del producto realizado. Además de facilitar el trabajo a los desarrolladores.

También se pudo comprobar que la utilización de un framework como base en el desarrollo de aplicaciones, trae la ventaja de estandarizar dicho desarrollo. Para la explicación de la importancia que tiene para cualquier proyecto mantener estandarizado su código, los autores estudiaron las ventajas y las desventajas de estandarizar el desarrollo de los productos. A continuación se exponen los resultados del estudio realizado.

1.5. Estandarización del software.

La palabra estándar tiene varios significados, pero en informática se define como: *“una serie de lineamientos técnicos detallados, destinados a establecer uniformidad en el desarrollo de programas (software) y compra de equipos (hardware).”* (20)

“El mundo cada vez se hace más pequeño. El hardware, software y el contenido cada vez son más independientes y no respetan fronteras internacionales. Esto incrementa cada vez más la necesidad del crecimiento común y la interoperabilidad.” (20)

El presidente del consejo de administración de la compañía Intel, Craig Barrett, en una entrevista sobre la estandarización, planteó que debido al creciente auge de las tecnologías, sin seguir ninguna norma universal de desarrollo, se hace prácticamente imprescindible la necesidad de la utilización de un estándar de software y hardware universal.

Barrett plantea también el paso de avance para el mundo de las tecnologías informáticas que sería utilizar estándares, al igual que lo fue en la electrónica, con las normas eléctricas; o en el cine, con las normas para la realización de películas.

“Cuando haya una norma común de protocolos, interfaces, y factores de forma, entonces la industria puede evolucionar en torno a las características comunes e innovar en la parte superior de ellos. Estas normas permiten a la industria avanzar en cada una de las empresas sin tener que hacer el terreno de la aplicación por su propia cuenta. Que ha sido el modelo para el éxito de la computadora personal, y ha sido el modelo para el éxito de electrónica de consumo, en gran medida. Debido a las normas, todo el mundo puede innovar y todo el mundo puede intercambiar. Las empresas pueden construir sus negocios, los consumidores pueden ampliar sus opciones, la tecnología avanza más rápido, y los usuarios obtienen más beneficios.” (20)

Producto a la necesidad de estandarizar los productos, en la actualidad la mayor parte de los proyectos de software exitosos tienen como cualidad común basar su trabajo alrededor de estándares de desarrollo de software. En algunos casos, utilizan estándares internacionales, y en otros crean e imponen sus propios productos como estándares. Como ejemplo, se puede mencionar el proyecto freedesktop.org que surgió por la motivación y visión de algunos desarrolladores de aplicaciones para escritorios, con el objetivo de buscar soluciones y tecnologías comunes, y así estandarizar el desarrollo de sus aplicaciones.

Otro ejemplo es la compañía de desarrollo de software Red Hat Inc.; una de las compañías de software libre y código abierto más exitosas. Se ha destacado por imponer sus soluciones y tecnología como estándares mundiales de desarrollo de software.

Se puede concluir que estandarizar el trabajo de los archivos de configuración del sistema, para estandarizar el desarrollo de las herramientas de configuración del sistema Nova, traería múltiples ventajas pues facilitaría el mantenimiento y la optimización de las mismas, además de suplir la carencia de aplicaciones para la configuración del sistema. El uso de un framework, que trabaje como capa abstracta entre la aplicación de configuración y los archivos, facilitaría dicha estandarización.

Entre los frameworks estudiados para el trabajo con archivos de configuración del sistema, está el framework utilizado por YaST para la gestión de sus archivos: SRC, conjunto de pequeñas aplicaciones que manejan los archivos por separado, es decir gestiona los archivos, en dependencia de su sintaxis.

Las tecnologías utilizadas por Linuxconf y Webmin, para gestionar los ficheros que necesitan, es similar al SRC. Pero estas herramientas son modulares, es decir trabajan por módulos. La interacción con los ficheros la realizan a través de módulos contenedores de funcionalidades especializadas para la gestión de archivos, las cuales varían en dependencia de la estructura y la sintaxis del fichero con el que se va a trabajar.

En ninguno de estos casos se estandariza la gestión de archivos de configuración del sistema, pues el trabajo con los mismos se realiza de forma independiente.

El otro framework estudiado es Augeas, que permite trabajar con los archivos de configuración del sistema sin interactuar con la sintaxis de estos.

La utilización de Augeas, para la gestión de archivos de configuración del sistema, permite agilizar el desarrollo de las aplicaciones que interactúan con dichos archivos. Además esta herramienta crea

una capa de abstracción entre el desarrollador y los ficheros del sistema, brindando un entorno estable de desarrollo, que facilita la implementación estándar de aplicaciones de configuración.

El respaldo de la comunidad de desarrollo de Augeas, posibilita que el desarrollo de dicha aplicación se mantenga en constante evolución, aumentando sus prestaciones y mejorando la calidad de sus funcionalidades. En la actualidad cuentan con la versión 0.5, que soporta 56 archivos distintos de configuración del sistema, aunque este número aumenta constantemente.

Por las características antes planteadas, que son los principales beneficios que aporta Augeas, se propone utilizar dicha herramienta en el sistema Nova para estandarizar la gestión de los ficheros de configuración, en el desarrollo de aplicaciones para la administración y configuración del sistema.

2.Augeas

Antes de la popularidad de GNU/Linux como sistema operativo de propósito general, se le conocía como un sistema operativo solo para expertos en computación ([Ver Anexo 2](#)). Por tanto no existía competitividad posible con los sistemas operativos de la Microsoft, ni Apple, pues tenían alto soporte de hardware y la configuración del sistema era sencilla de realizar.

Dada las circunstancias, la comunidad y grandes compañías ([Ver Anexo 3](#)) se han dado a la tarea de enfrentar estos problemas, dedicando un gran esfuerzo para dotar al kernel de Linux del mayor soporte de hardware posible, y por supuesto de herramientas personalizadas que interactúen con él. Debido a esto, los propios desarrolladores se han dado cuenta que es muy difícil construir un gran número de herramientas dada la diversidad de ficheros con los que interactúan, el costo de tiempo y el esfuerzo que lleva cada una de ellas. Producto de la carencia de herramientas que facilitaran el trabajo con ficheros, surgió la necesidad de crear conjuntos de librerías, módulos, o en algunos casos frameworks, algunos ejemplos de estas tecnologías fueron descritas en el capítulo anterior.

En el caso de las herramientas que manipulan archivos de configuración del sistema, los mayores aportes los tiene YaST con su clase SRC, y LinuxConf con un conjunto de librerías para la gestión de archivos.

A su vez, Harmony, proyecto que surgió con el objetivo de crear un lenguaje de programación donde los programadores puedan traducir una información específica en un modelo cualquiera, y que ese proceso sea reversible, desarrolló Boomerang. El cual basa su trabajo en la traducción de la información en la estructura abstracta, árbol, y viceversa.

Al no existir una herramienta común para cualquier distribución de GNU/Linux que facilite de manera considerable la gestión de ficheros de configuración. En el 2007, David Lutterkort, desarrollador de Red Hat; crea a Augeas, herramienta de edición de ficheros de configuración del sistema GNU/Linux, utilizando como núcleo a Boomerang.

El nombre de esta herramienta tiene sus orígenes en la mitología griega; donde Áugeas o Augías (griego antiguo, Augeías 'brillante') era un rey de Élide, hijo del dios-Sol Helios y de Naupidame.

Augeas lanza su versión 0.0.1 en diciembre del 2007; y a partir de ese momento ha alcanzado grandes mejoras en cada nueva versión. Lanzando una nueva versión mensualmente, y en

ocasiones más de una. En mayo del 2008 lanzan la 0.1 y en la actualidad cuentan con la versión 0.5.

El desarrollo vertiginoso de dicha herramienta, se debe en gran medida al apoyo que le ha dado la comunidad de software libre, pues actualmente cuenta con un gran número de contribuyentes, entre los que se destacan desarrolladores de Red Hat, Fedora, Debian y Ubuntu. También cuenta con mantenedores de otras distros importantes como SuSe y Gentoo.

Como muestra del constante mejoramiento de la herramienta se puede decir que actualmente el código fuente cuenta con aproximadamente 1000 revisiones, que equivale a 2.1 revisiones por día; cifra importante para una herramienta relativamente pequeña.

Augeas es totalmente libre, está resguardada por la licencia GPL (GNU General Public License) en su versión 2.1, aunque también soporta LGPL (GNU Lesser General Public License), para los desarrolladores de aplicaciones no libres.

2.1. ¿Qué es Augeas?

Augeas es un editor que analiza los archivos en sus formatos nativos y los transforma en un tipo abstracto de datos llamado árbol, y viceversa.

La manipulación de la configuración la realiza mediante este árbol, y los cambios que sean aplicados a él se guardan de forma nativa en dichos archivos de configuración. Para su realización utiliza unas estructuras llamadas Lenses (en español lentes), que permiten que Augeas pueda entender cualquier archivo de configuración.

Es una herramienta modular, construida con el lenguaje de programación C (Ver [Anexo 4](#)), brindando rapidez y eficiencia desde el punto de vista del uso de recursos computacionales.

“Augeas cuenta con una API de no más de una docena de funcionalidades, que le brindan a los desarrolladores la posibilidad de interactuar con la herramienta. También brinda implementaciones en Python, Ocaml, Ruby, Perl, Haskell, Java, entre otros. Permite además la interacción mediante la consola de Linux, a través de la herramienta augtool.” (19)

Al igual que la mayoría de los proyectos de software desarrollados en la actualidad, tiene definida una arquitectura en la cual basa su implementación.

La arquitectura de software tomó mucha preponderancia en los últimos años, como algo clave a la hora de diseñar una solución informática. Pues con el transcurso de los años, la complejidad y

tamaño de los sistemas software se fue incrementado, por tanto, la capacidad para responder rápidamente ante los cambios y optimizar los procesos es un factor clave para la competitividad y el crecimiento de las organizaciones.

Además le brinda estabilidad y calidad al producto, pues al poner en práctica las definiciones de la arquitectura y del diseño, se evitan muchos errores.

Para acoplar esta herramienta en el sistema Nova, fue necesario el estudio detallado de su arquitectura y diseño.

2.2. Arquitectura y Diseño de Augeas.

Para darle respuesta a los diferentes procesos de desarrollo, se han desarrollado distintos estilos arquitectónicos. Algunos ejemplos de estos son:

- Filtro-Tubería
- Sistemas Orientados a objetos (Inc. TAD)
- Invocación Implícita (Basada en Eventos)
- Sistemas basados en Capas
- Sistemas basados en repositorios
- Máquina Virtual o Intérprete (21)

A continuación se explicará en qué consiste el estilo arquitectónico basado en capas, pues el desarrollo de Augeas, está basado sobre dicho estilo.

2.2.1. Estilo arquitectónico en capas.

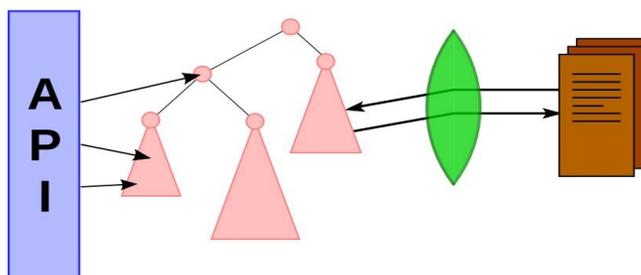
La arquitectura de software en capas, se está utilizando mucho en la actualidad, por las amplias ventajas que tiene. *“Los principios de las definiciones de la misma aparecieron en 1968, con Edsger Dijkstra, quien sugería seguir pasos formales para descomponer problemas mayores, Dijkstra fue uno de los introductores de la noción de sistemas operativos organizados en capas que se comunican sólo con las capas adyacentes y que se superponen “como capas de cebolla.” (22)*

El diseño de aplicaciones en capas es ideal para la creación de sistemas adaptables, donde cada componente puede ser utilizado y reutilizado en nuevas combinaciones para satisfacer requisitos de negocio dinámicos.

Esto permite a los desarrolladores y a las nuevas aplicaciones reutilizar componentes. En un entorno cambiante como el actual, utilizar aplicaciones basadas en diseños de capas, posibilita a los proyectos ser más ágiles y adaptables en proporcionar productos a sus clientes. (Ver [Anexo 5](#))

2.2.1.1. Descripción del diseño por capas.

Augeas basa su trabajo en 4 capas las cuales se describen a continuación. La descripción de cada capa de hará de izquierda a derecha con el objetivo de seguir un orden lógico.



Los Figura 1. Describe la arquitectura de Augeas

- Capa de Interfaz

Permite la interacción con Augeas, conforma la API del sistema. A continuación se explicará de forma detallada en qué consisten cada una de las funciones que brinda la herramienta.

Nota: las funciones se describen en el lenguaje de programación nativo de la herramienta, o sea en el lenguaje C.

- `aug_init`: Inicia la librería, creando el árbol que contiene las configuraciones que se describen en cada módulo (en cada Lens).
- `aug_close`: Destruye el árbol de configuraciones.
- `aug_get`: Obtiene los datos dado un fichero o un atributo de una configuración.
- `aug_set`: Agrega un nuevo valor a un atributo o un nuevo atributo.

- `aug_insert`: Permite agregar un nuevo nodo al árbol.
- `aug_rm`: Permite que el desarrollador pueda eliminar tanto un atributo como un nodo del árbol de configuración.
- `aug_mv`: Brinda la posibilidad de moverse dentro del árbol.
- `aug_match`: Verifica la existencia tanto de un nodo, como de un fichero, o también de un atributo en específico.
- `aug_save`: Si se ha modificado, salva la configuración en el lenguaje de programación nativo. En caso de que se haya especificado en el `aug_init` que se va a crear una salva de la antigua configuración, sobrescribe la configuración y la herramienta guarda una salva. En caso de especificar que se cree una nueva, mantiene intacta la configuración y crea un nuevo archivo con la configuración modificada.
- `aug_print`: Imprime la información solicitada.

- Capa de Estructura:

En esta capa es donde se concentra toda la construcción del árbol de configuración. Cada nodo del árbol está estructurado de la siguiente manera.

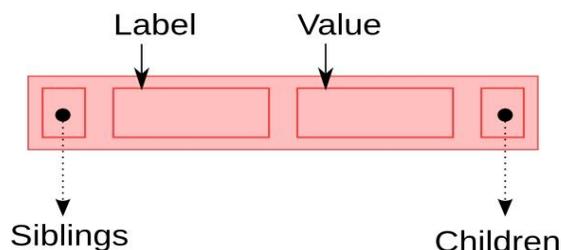


Figura 2. Describe la estructura de un nodo de Augeas

Cada nodo contiene dos punteros, el primero apunta al nodo siguiente en caso de ser este un nodo que esté a su mismo nivel. El segundo puntero se usa en el caso de que el nodo en cuestión se comporte como un nodo padre, o sea que tenga un nivel inferior a él. El nodo también está compuesto por dos espacios de memoria que son usados para:

1. Guardar el nombre del atributo.
2. Guardar el valor del atributo.

A continuación se muestra una figura que permite ver claramente la estructura de un árbol de Augeas, con sus valores.

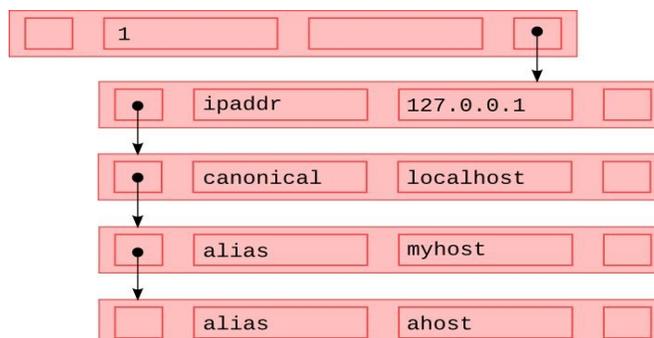


Figura 3. Árbol ejemplo de una configuración del fichero /etc/hosts.

- Capa de Parseo:

Posiblemente en esta capa es donde se encuentra lo más novedoso que aporta Augeas en el mundo de las tecnologías de procesamiento de archivos. En esta se encuentran los módulos, que no son más que Lenses.

Los Lenses son los encargados de brindar soporte a cada fichero de configuración, pues cada uno describe un archivo. Por tanto la potencia y utilidad de Augeas es directamente proporcional al número de Lenses que soporte y por supuesto a la calidad con que estos fueron desarrollados. Para no romper con el orden lógico, se dedicará el siguiente epígrafe al mejor entendimiento de los Lenses.

- Capa de Persistencia o Ficheros:

En esta capa radican los ficheros de configuración soportados por Augeas. Un fichero de configuración no es más que un archivo que forma parte de una aplicación, y especifica ciertos valores necesarios para que la aplicación funcione correctamente y sea configurable. Dichos ficheros son extremadamente complejos de manipular, por el hecho de la gran diversidad de estos según sus sintaxis y la información que manejan.

2.3. Módulos de Augeas: los Lenses.

Como se mencionó anteriormente Augeas es una herramienta modular, pero a diferencia de otras herramientas modulares esta depende totalmente de sus módulos, ya que son los encargados de permitir la transformación de un fichero en árbol y viceversa. A estos módulos se le llaman Lenses.

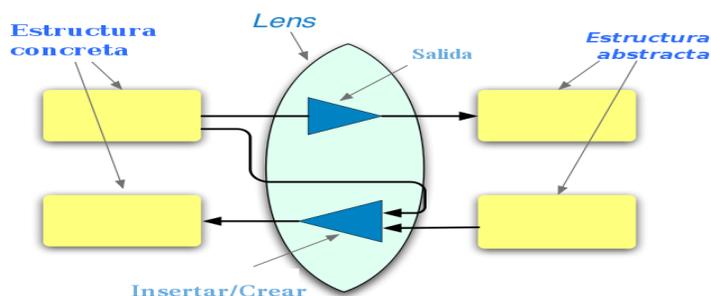


Figura 4. Describe el funcionamiento de los Lenses.

En la figura 4 se describe el funcionamiento de los Lenses, se muestra como a través de estas estructuras existe la transformación de una estructura concreta. En el caso de Augeas sería un archivo de configuración, en un árbol; característica principal de los lenguajes bidireccionales.

2.3.1. Programas bidireccionales.

“La necesidad de editar datos a través de un modelo se plantea en una serie de aplicaciones en diferentes áreas de la informática. Sin embargo, dado que la mayoría de los sistemas sólo tienen un apoyo limitado de vistas actualizables, por lo general tienen que ser aplicados mediante dos programas separados, uno para calcular la opinión de la fuente, y otro para manejar las actualizaciones. Este diseño hace a la programación rudimentaria, tediosa, difícil de razonar, y crea una pesadilla a la hora del mantenimiento.

A diferencia de los programas comunes, que sólo funcionan en una dirección, los programas en un idioma bidireccional se pueden ejecutar tanto hacia adelante como hacia atrás.” (23)

Cuando estos programas leen la información de izquierda a derecha crean un mapa abstracto que

puede ser manipulado, y al ser leída la información de derecha a izquierda permite que sean restauradas y eliminadas las ambigüedades. Los programas bidireccionales fueron diseñados con la intención de que la construcción de modelos de datos y modelos de vista no sea una tarea tediosa, y que la transformación entre modelos fluya correctamente. En el caso de Augeas el modelo de datos lo conforman: las variables y funciones correspondientes a las propiedades y sintaxis de los ficheros de configuraciones. Y el modelo vista sería el árbol. A continuación se presenta una lista de productos que usa la programación bidireccional como eje central de su existencia.

- Boomerang: Es un lenguaje de programación para la escritura de Lenses, que se utiliza para la transformación de estructuras de textos.
- PADS: Sistema para el procesamiento de datos ad hoc.
- PSDLab: Dependencia para el procesamiento de XML.
- XSugar: Lenguaje XML, para sintaxis en dúos.
- Augeas: Herramienta para la edición de ficheros de configuración.

Para la creación de un Lens, Augeas realiza una serie de pasos que se explicarán a continuación.

2.3.2. ¿Cómo crear módulos para Augeas?

El trabajo con los módulos está dividido en cinco grandes grupos:

- Creación de los Lenses: Comienza por crear el Lense de prueba, necesario para probar el módulo que se está desarrollando. Seguido a esto, se crea el módulo y se prueba usando *augparse*, una herramienta incluida dentro de Augeas con el objetivo de facilitar y propiciar la creación de estos módulos.
- Trabajo con los comentarios: Trabaja con los comentarios de los ficheros de configuración.
- Módulos comunes: Augeas permite reutilizar módulos comunes, o sea se pueden utilizar módulos ya existentes para la creación de nuevos Lenses.
- Módulos genéricos: La herramienta cuenta con una lista de módulos genéricos, pero también permite que los desarrolladores puedan agregar otros según sus necesidades.
- Documentación del módulo: Augeas usa NaturalDocs, para generar la documentación desde los comentarios.

Para la creación de un Lens óptimo, es recomendable trabajar con cada uno de los grupos. Aunque se pueden utilizar solo los dos primeros grupos.

2.4. Ideas futuras del proyecto.

Como Augeas es un proyecto en desarrollo con una comunidad activa, cuenta con algunas líneas de trabajos futuras. A continuación se describen algunas de estas ideas:

- **Mejorar soporte de archivos:** Mejorar los archivos de configuración de Augeas, y mejorar la forma en que están representados en el árbol.
- **GUI:** Creación de una interfaz gráfica para la edición de los archivos de configuración.
- **Generar esquemas:** Generar los esquemas de la descripción de los árboles de Augeas.
- **Migración a Augeas:** Adaptar las aplicaciones de configuración, para que usen como backend a Augeas.
- **Extensiones del lenguaje:** Mejorar Augeas como lenguaje ML.
- **Servicio Dbus:** Construir un servicio de Dbus basado en Augeas e integrado a PolicyKit, que permita a los usuarios configurar archivos según las políticas de seguridad.
- **Expresión Regular:** Escribir una expresión regular en la librería autómatas finitas de Augeas que se base en derivaciones. Mejorando el tiempo de ejecución de esta librería.

Es importante señalar, que esta lista de tareas por hacer que presentan los desarrolladores de la herramienta muestra la salud con que cuenta este proyecto.

Se puede concluir que Augeas a pesar de ser una herramienta que lleva aproximadamente año y medio de desarrollo, cuenta con funcionalidades de gran utilidad para la gestión de archivos de configuración de sistemas GNU/Linux. Además está basada sobre una arquitectura flexible, permitiendo la reutilización y expansión de su código.

La estructura que utiliza Augeas para el trabajo con la información de los archivos, facilita el estudio y la utilización de la herramienta. Propiedad necesaria para la inserción exitosa de dicha herramienta en aplicaciones que gestionen archivos de configuración del sistema operativo Nova.

3. Augeas en Nova

En el presente capítulo se va a demostrar a través de las aplicaciones Capoeira y EcumeniX, las facilidades que brinda la utilización de Augeas para gestionar los ficheros de configuración que interactúan con las mismas. Usando esta demostración como caso de estudio para estandarizar el trabajo con los ficheros de configuración en las aplicaciones desarrolladas por el proyecto Nova. Siendo estas las herramientas de configuración que se utilizan en dicho sistema.

3.1. Sistema Operativo Nova

Nova es una distribución de GNU/Linux orientada a Escritorio desarrollada en la Universidad de las Ciencias Informáticas. Inicialmente surgió con el objetivo de facilitar la migración a la facultad de software libre de la UCI. Pero la distribución continuó creciendo como sistema, y en estos momentos tiene como principal objetivo el desarrollo y mantenimiento de un sistema operativo que responda a las necesidades del país.

El crecimiento de Nova ha desencadenado el desarrollo de varias aplicaciones, para suplir sus necesidades y aumentar la calidad del sistema. Entre los principales proyectos que se están llevando a cabo en el proyecto Nova están:

1. Guano, Entorno de Escritorio Ligerero.
2. Summon, Instalador de Aplicaciones.
3. Serere, Instalador de la Distribución Nova.
4. Panel o Centro de Control de Nova.
5. Capoeira y EcumeniX, Integración con Microsoft Windows.

La necesidad de facilitar y agilizar el desarrollo de las herramientas mencionadas ha demostrado que se debe cambiar la forma de gestionar los archivos que interactúan con estas; pues el modo tradicional (utilizando módulos especializados del lenguaje que se esté usando para implementar la aplicación, o librerías, como Glib) convierte la implementación de la aplicación en un trabajo muy complejo, además dificulta el desarrollo estándar de aplicaciones. Para demostrar las ventajas de la utilización de Augeas en estos casos, se migró la gestión de ficheros de Capoeira y de EcumeniX,

utilizando dicha herramienta.

A continuación se explica la diferencia entre la implementación de Capoeira con la utilización de Augeas como capa de abstracción para la gestión de ficheros y utilizando funciones nativas de Python (lenguaje de desarrollo utilizado) para la gestión de archivos de configuración.

3.2. Casos de estudio

3.2.1. Capoeira

Capoeira es una herramienta visual integrada al explorador de ficheros de Gnome (*Nautilus*), que permite compartir carpetas a través del servicio Samba usando el menú contextual asociado a cada carpeta cuando se oprime el *click* derecho (o botón derecho del ratón) sobre ella. Utilizando Capoeira no es necesario trabajar directamente con el fichero de configuración de Samba, *smb.conf*, pues una de las tareas de esta aplicación es configurarlo con las opciones de seguridad requeridas por el usuario. Además establece los permisos necesarios para el sistema de archivos subyacente, haciéndolos compatibles con los declarados en el fichero de configuración de Samba. Debido a que los permisos típicos de Unix, comúnmente conocidos como permisos UGO (usuario, grupo, otros), no son suficientes para garantizar que los permisos de Samba tengan sus homólogos en el sistema de archivos, Capoeira usa un sistema de permisos que se viene implementando hace ya algún tiempo en las distintas variantes de Unix: las Listas de Control de Acceso de Posix (Posix ACL).

La aplicación está implementada en los lenguajes de programación Python y C, utilizando C fundamentalmente en las optimizaciones. Esta cuenta con facilidades de uso como búsqueda con completamiento automático de nombres de usuario tanto locales como del dominio, interfaz para visualizar todas las carpetas compartidas (Ver [Anexo 6](#)) y acceder a sus opciones, límite de conexiones por carpeta, control de visibilidad de las carpetas, deshabilitar/habilitar compartidos, manejo sencillo de los *smbpasswords*, y otras.

Esta herramienta trabaja directamente con los ficheros de configuración *smb.conf*, y *fstab* para montar los directorios compartidos. Entre las funcionalidades que utiliza para gestionar la configuración de estos ficheros están principalmente *obtener* y *modificar* la información que contienen.

A continuación se explicará como gestiona el trabajo con estos archivos y cuáles son las ventajas

obtenidas de la migración de dicha herramienta a Augeas.

3.2.1.1. Gestión de ficheros de configuración en Capoeira.

Capoeira, como se explicó en el epígrafe anterior está desarrollado con el lenguaje Python y para gestionar los archivos de configuración con los que interactúa se utilizaron funcionalidades nativas de dicho lenguaje de programación. Entre las más utilizadas son: *open* para abrir el fichero, *close* para cerrarlo, *readline* para leer una línea, *write* para escribir, las estructuras de datos como las *listas* para guardar la información del archivo, entre muchas otras.

Gestión de smb.conf

Entre los métodos implementados en Capoeira para la gestión del fichero de configuración del archivo smb.conf están:

get_section_position → Se encarga de informar la posición de la sesión que le pasan por parámetro.

unshare → Elimina la carpeta compartida, es decir la sesión y sus propiedades.

share → Crea una nueva sesión, con las propiedades de la carpeta que se va a compartir.

read_shares_data → Lee los datos de la sección que se le pasa por parámetro.

list_shares → Lista las carpetas compartidas.

get_domain_prefix → Busca si la computadora está compartida en el dominio y el prefijo de la misma.

set_property_by_section → Inserta la propiedad que se le pasa por parámetro, en la sesión que se le indique.

remove_property_by_section → Elimina la propiedad que le pasan por parámetro en la sesión indicada.

get_property_by_section → Muestra la información de la propiedad que le pasan por parámetro, dentro de la sesión que se le indica.

find_property_by_section → Encuentra la propiedad que se le indica dentro de la sesión pasada por parámetro.

Estos métodos fueron implementados con las funcionalidades nativas del lenguaje Python. A través

de ellos Capoeira le brinda al usuario la posibilidad de configurar el servicio Samba, sin tener previo conocimiento de la sintaxis del archivo *smb.conf*. El desarrollo de cada uno de dichos métodos trae consigo el crecimiento de la complejidad del trabajo para el desarrollador, pues tiene que interactuar directamente con el archivo. En cambio si para la implementación de la gestión del archivo *smb.conf* se utilizara Augeas la implementación de la aplicación se convertiría en una tarea más sencilla, como ejemplo para explicar la diferencia que existe entre: implementar los métodos mencionados utilizando la herramienta propuesta para la gestión de archivos, y sin el uso de la misma, se tomó la función *list_shares*.

En el [Anexo 7](#) aparecen dos diagramas de bloque del método *list_share*, el de la izquierda representa los pasos que se llevan a cabo utilizando para la gestión del archivo *smb.conf*, las funcionalidades que tiene Python para el trabajo con archivos; y la imagen de la derecha representa al mismo método, pero utilizando Augeas para la gestión del archivo.

Este método se encarga de listar las carpetas compartidas, para esto, inicialmente crea una lista vacía (*share_list*) y una variable (*length*) que toma la cantidad de líneas que tiene el archivo. Luego entra en un ciclo, hasta que recorre todas las líneas del fichero. Cada vez que entra en un directorio compartido, toma los valores de este, es decir las propiedades que tiene, luego guarda las propiedades en la lista (*share_list*) y al final retorna *share_list*.

La complejidad de este método es encontrar cada directorio compartido y guardar los valores de sus propiedades. Que es precisamente donde entra el trabajo de Augeas, creando una capa entre el desarrollador y el archivo, para facilitar la gestión del mismo durante la implementación.

Como se puede apreciar con la utilización de Augeas el método se hace más sencillo de implementar, posibilitando agilizar el trabajo del desarrollador.

Gestión de *fstab*.

El fichero *fstab* (*file systems table*), es el encargado de especificar cómo montar cada dispositivo y cuales opciones de montaje utilizar al iniciarse el sistema. El trabajo con el archivo *fstab* tiende a ser complejo si no se estudia previamente su sintaxis, al igual que pasa con la mayoría de los archivos de configuración de los sistemas GNU/Linux.

Con la configuración de este fichero la herramienta posibilita el acceso rápido a los directorios compartidos. Para gestionar el *fstab* Capoeira también se apoya en las funcionalidades de Python, y entre los métodos que define para implementar dicha tarea están:

get_fs_data_for_path → Obtener información.

binded_mountpoint_exists → Obtener los puntos de montaje que existen.

add_entry → Adicionar el directorio que se desea compartir.

remove_entry → Eliminar el directorio que se desea remover de la lista de ficheros compartidos.

set_proper_options → Verifica que los directorios compartidos tengan la opción ACL, en caso de no tenerla se le agregan y devuelve una lista con estos directorios modificados.

is_fs_acl_capable → Verifica si el sistema de archivo soporta los permisos ACL o no.

Para explicar mejor la diferencia existente entre la implementación de estos métodos, utilizando Augeas para la gestión del archivo y la implementación de los mismos usando las funciones nativas de lenguaje de programación seleccionado para su desarrollo; se escogió el método *get_fs_data_for_path* para detallar los pasos de su implementación. Y de esta forma visualizar las ventajas de la utilización de Augeas.

En el [Anexo 8](#) aparecen dos diagramas de bloque del método *get_fs_data_for_path*, el de la izquierda representa los pasos que se llevan a cabo utilizando para la gestión del archivo *fstab*, las funcionalidades que tiene Python para el trabajo con archivos. Y la imagen de la derecha representa al mismo método, pero utilizando Augeas para la gestión del archivo.

Este método se encarga de buscar y devolver los dispositivos con sus propiedades para montaje, que no tengan la opción "bind". Esta opción permite montar dispositivos que ya estaban montados. Para esto el método crea una lista vacía y entra en un ciclo que recorre todas las líneas del archivo, verificando en cada una si existe dicha opción; y retorna la lista con los dispositivos que no tengan a "bind" como opción.

A continuación se describe la migración de la gestión de los ficheros que interactúan con la herramienta EcumeniX a Augeas.

3.2.2. EcumeniX

EcumeniX es una herramienta orientada a administradores de sistemas y a usuarios finales de escritorio. Su principal función es unir una computadora con sistema operativo GNU/Linux a un dominio controlado por un *Directorio Activo de Microsoft (Microsoft Active Directory)*.

La interfaz gráfica que EcumeniX ofrece al usuario de escritorio está construida con GTK (Ver [Anexo 9](#)), y consiste en un sencillo asistente que detecta de manera automática muchas de las informaciones esenciales del dominio, y deja por parte del usuario la entrada del nombre del dominio al que se desea unir y un nombre de usuario y contraseña válidos, que tengan permisos en el controlador de dominio para unir máquinas al mismo. Por otra parte, si el usuario es administrador de un servidor que no tiene instalada interfaz gráfica de usuario, tiene la posibilidad de usar un script que permitirá de la misma forma, unir rápidamente su servidor al dominio, especificando un conjunto mínimo de datos y acciones a tomar, que le permitirá brindar todos los servicios que desee utilizando el sistema de autenticación del Directorio Activo.

Al igual que Capoeira esta herramienta fue desarrollada utilizando el lenguaje de programación Python. Su forma de automatizar todo el proceso trata de acercarse lo más posible a la metodología KISS (Keep It Simple Stupid), en otras palabras, mientras más sencillo, mejor. Aunque utiliza para la gestión de los archivos de configuración con que interactúa, las funcionalidades que tiene Python para dicha tarea, que como se explicó en el epígrafe anterior dificulta el desarrollo ágil de la aplicación.

EcumeniX interactúa con los siguientes archivos de configuración:

- *hosts* → Tiene definido los dominios de los ip que se utilizan.
- *kerberos* → Kerberos es un moderno sistema de autenticación de red basado en la idea de expedir un ticket a un usuario una vez que se ha autenticado.
- *ntp.conf* → Guarda la configuración horaria.
- *pam* → Es un conjunto de librerías compartidas que habilitan al administrador local del sistema para escoger el modo en que los programas autenticarán a los usuarios.
- *nss* → Conjunto de librerías compartidas que habilitan al administrador local del sistema para escoger el modo en que los programas autenticarán a los usuarios.
- *smb.conf* → Configura Samba.

El caso de estudio de esta herramienta se centró en los archivos *hosts* y *ntp.conf*.

Gestión de hosts.

Para gestionar el fichero *hosts*, se creó *hostsparser.py* que hereda del fichero *confparser.py*. El archivo *confparser.py* contiene las funcionalidades básicas para trabajar con los ficheros.

En *hostsparser.py* está definida la función *transform*, que se utiliza para agregar las direcciones IP de los servidores utilizados para la conexión y el IP del anfitrión; con sus respectivas direcciones canónicas en cada caso.

Para obtener la información necesaria, la función inicia con un ciclo que recorre todas las líneas del archivo, verificando si existe el IP especificado y en caso de encontrarlo lo elimina. Luego verifica si hay alguna línea vacía y adiciona los datos de los IP pasados por parámetro. En el [Anexo 12](#) aparecen dos diagramas de bloque del método *transform*, el de la izquierda representa los pasos que se llevan a cabo utilizando para la gestión del archivo *hosts*, las funcionalidades que tiene Python para el trabajo con archivos. Y la imagen de la derecha representa al mismo método, pero utilizando Augeas para la gestión del archivo.

Gestión de ntp.conf.

La gestión de este archivo se realiza a través del fichero *ntpparser.py*, que hereda al igual que *hostsparser.py* de *confparser.py*. El archivo *ntpparser.py* contiene la función *transform* que se utiliza para escribir nuevos servidores ntp. La función antes verifica la existencia de servidores, y en caso de existir alguno, lo elimina.

El método inicialmente entra en un ciclo donde recorre el archivo *ntp* verificando en cada línea la existencia de la palabra “*server*”, en caso de existir alguna elimina la línea. Al salir del ciclo agrega los servidores especificados por parámetro. En el [Anexo13](#) aparecen dos diagramas de bloque del método *transform*, el de la izquierda representa los pasos que se llevan a cabo utilizando para la gestión del archivo *ntp*, las funcionalidades que tiene Python para el trabajo con archivos; la imagen de la derecha representa al mismo método, pero utilizando Augeas para la gestión del archivo.

Conclusiones

El estudio realizado permitió adoptar una herramienta que da respuesta a la problemática que enfrentan los desarrolladores durante la implementación de aplicaciones que interactúan con archivos de configuración del sistema GNU/Linux.

Estandarizar la gestión de los archivos de configuración del sistema Nova, brinda facilidades para el mantenimiento de las aplicaciones que mantienen interacción con los mismos. Y permite asegurar la calidad del trabajo con dichos ficheros.

La utilización de Augeas como herramienta para la gestión de los archivos, permite que el desarrollador no interactúe directamente con los ficheros de configuración; incidiendo en la calidad y rapidez del desarrollo de aplicaciones de configuración; pues le ofrece al programador una capa de abstracción para el trabajo con estos archivos.

Los resultados obtenidos en las aplicaciones migradas hacia Augeas, pueden extenderse a la mayoría de los proyectos que necesiten interactuar con archivos de configuración de sistemas GNU/Linux.

Recomendaciones

Se recomienda la utilización de Augeas para gestionar los archivos de configuración del sistema Nova, durante la implementación de herramientas de configuración requeridas por dicho sistema.

Se recomienda migrar progresivamente las aplicaciones desarrolladas en el proyecto Nova, que interactúen con archivos de configuración de sistema, hacia Augeas.

Se recomienda que los desarrolladores se integren a la comunidad de Augeas, para apoyar el desarrollo de la herramienta, tanto en la creación de nuevos Lenses, como en la implementación de nuevas ideas o funcionalidades.

Se recomienda profundizar el estudio, junto a la comunidad de Augeas para el desarrollo de una GUI, que facilite incluir soporte para nuevos archivos de configuración a Augeas.

Referencias Bibliográficas

1. **Climent, Jesús y García, José Marcos Chalmés.** *¿Por qué cambiar a LINUX?* [En línea] <http://personales.alumno.upv.es/~jochagar/canvi.html#toc5>.
2. **Ferré, Xavier.** Usabilidad: Software pensado para los usuarios. *Universia*. [En línea] http://www.universia.es/portada/actualidad/noticia_actualidad.jsp?noticia=90993.
3. LINUXCONF. *Solucorp*. [En línea] <http://www.solucorp.qc.ca/linuxconf/>.
4. YaST. *openSUSE*. [En línea] 29 de dic de 2007. <http://es.opensuse.org/YaST>.
5. **Thomas Yeung, Stanislav Visnovsky.** Building YaST Modules.
6. **Srain, Jin.** *YaST Architecture*.
7. WEBMIN. [En línea] <http://www.webmin.com/>.
8. K Desktop Environment. *KDE The Application Development Framework*. [En línea] <http://www.kde.org/whatiskde/>.
9. GNOME: *The Free Software Desktop Project*. [En línea] <http://www.gnome.org/>.
10. Package: kadmin (4:3.5.9-2) . *Debian*. [En línea] <http://packages.debian.org>.
11. kadmin-3.1.4. *Más Allá de Linux From Scratch: Versión 5.0*. [En línea] <http://www.escomposlinux.org/lfs-es/blfs-es-5.0/kde/kde-admin.html>.
12. CPAN. [En línea] 9 de febrero de 2009. <http://cpan.org/>.
13. **García, Javier, y otros.** *Aprenda Java como si estuviera en primero*. San Sebastian : s.n., 1999.
14. **Maldonado, Daniel M.** Py Qt Desarrollando Aplicaciones De Escritorio El Co Di Go K. *slideshare*. [En línea] <http://www.slideshare.net/juancabicho/py-qt-desarrollando-aplicaciones-de-escritorio-el-co-di-go-k-presentation>.
15. QFileInfo Class Reference. *Qt*. [En línea] 2009. <http://doc.trolltech.com/3.2/qfileinfo.html>.
16. Biblioteca de documentación de GNOME. *GLib Reference Manual*. [En línea] 2008. <http://library.gnome.org/devel/glib/>.
17. **Markiewicz, Marcus Eduardo y Lucena, Carlos J.P.** acm.org. *El Desarrollo del Framework Orientado al Objeto*. [En línea] <http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html>.
18. **Robert, Don y Johnson, Ralph.** *Patterns for evolving frameworks*. 1997.
19. **Baptiste, Juan Luio.** Kde4 Se Libre. [En línea] <http://www.slideshare.net/campuspartycolombia/kde4-se-libre>.
20. Augeas. *Meta Admin*. [En línea] <http://meta-admin.blogspot.com/2008/04/augeas.html>.
21. **Barrett, Craig.** Executive Interviews: Craig Barrett. *Intel*. [En línea] <http://www.intel.com/standards/execqa/qa0904.htm>.
22. **Cuesta, Carlos E.** *Arquitecturas de Software*.
23. **Martín, Yanet Espinol.** Arquitectura de software. Arquitectura orientada a servicios. *Ilustrados.com, Comunidad educativa mundial*. [En línea] julio de 2008. <http://www.ilustrados.com/publicaciones/EkEuAuVEyISfYPhaVc.php>.

Bibliografía

- “Adept - Wikipedia, la enciclopedia libre”; <http://es.wikipedia.org/wiki/Adept>.
- **Wesley, Addison.** *Desing for Object-Oriented Software Development*. 1995.
- **Villate., Jaime E.** Introducción al XML. [En línea] 2001.
- **Telechea, Fernando y Matosas, Germán.** *Frameworks de Aplicación*. 2006.
- **TechTarget.** The leading IT Encyclopedia and learning center. [En línea] http://whatis.techtarget.com/definition/0,289893,sid9_gci1103696,00.html.
- **Srain, Jin.** YaST Architecture. 2006.
- **EL FUTURO DEL DESARROLLO DE SOFTWARE ORIENTADO A ASPECTOS. Software, Grupo de Investigación en Ingeniería de.** 2005, Tertulia de Ingeniería de Software.
- **Schmuller, Joseph.** *Aprendiendo UML en 24 horas*.
- **Rumbaught, James.** *UML Manual de Referencia*. 2000.
- **Prieto, Felix, y otros.** *Construccion de frameworks basada en analisis de conceptos formales y soportada por Mecanos*.
- **Pressman, Roger S. y Ince, adaptado por Darrel.** *Ingeniería del Software Un enfoque práctico*. Madrid: s.n., 1997.
- **Penas, Juan José Sánchez.** *Desarrollo de software libre de gestión empresarial*. Buenos Aires, Argentina: s.n., 2006.
- **Montaña, Emilio González.** SW Saber. *Scrum*. [En línea] 2008. <http://swsaber.com/scrum/>.
- **Jordi Mas, David Megías Jiménez, Marc Gibert Ginestà, Álvaro Peña González.** *Software libre: Ingeniería del software en entornos de SL*. Marzo 2005.
- **Jacobson, Ivar, Booch, Grady y Rumbaught, James.** *El Proceso Unificado de Desarrollo de Software*. 2000.

- **Hugo, Arboleda Jimenes.** Modelos de ciclo de vida en desarrollo de software. [En línea] Septiembre de 2005. <http://www.acis.org.co/index.php?id=551>.
- **Hoffstat, Arturo.** <http://www.arturo.hoffstadt.cl/wp/category/kde>. [En línea]
- **GONZÁLEZ, CARLOS OMAR TORRES.** *INGENIERÍA DE DOMINIO ORIENTADA A OBJETOS APLICADA AL DESARROLLO DE UN FRAMEWORK PARA LA IMPLEMENTACIÓN DE AGENTES DE ADMINISTRACIÓN DE RED.* INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY, s.l.: 2003.
- **Gomez.Omar.** *Bases conceptuales de frameworks y su importancia para lograr reutilización.*
- **Fírvida, Abel y Rodriguez, Adisleydis.** *Guano. Entorno de escritorio libre y de código abierto cubano.* Ciudad de la Habana: s.n., 2008.
- **Canós, José H., Letelier, Patricio y Penadés, Ma Carmen.** *Métodologías Ágiles en el Desarrollo de Software.*
- **What are the GNOME System Tools?** [En línea] <http://projects.gnome.org/gst/>.
- **Wapedia.** [En línea] 14 de enero de 2009. [Citado el: 24 de enero de 2009.] <http://wapedia.mobi/es/GLib>.
- **Quienes Somos. Nova.** [En línea] 2009. <http://www.nova.uci.cu/website/>.
- **Python.** *Style Guide for Python Code.* [En línea] 2009. <http://www.Python.org/dev/peps/pep-0008/>.
- **msdn.** *Inicio de .NET Framework.* [En línea] Microsoft Corporation., 2009. <http://msdn.microsoft.com/es-es/netframework/default.aspx>.
- <http://www.seas.upenn.edu/~harmony/old/>. [En línea] [Citado el: 2009 de enero de 20.] WWW.LINUX-MAGAZINE.ES.
- **Linuxconf .** [En línea] <http://www.solucorp.qc.ca/linuxconf/>.
- **GConf, el sistema de configuración.** *Programación en el entorno GNOME.* [En línea] <http://webs.ono.com/gonav/librognome-html/c11045.html>.
- **GConf.** *Gnome Chile.* [En línea] <http://wiki.gnome.cl/GConf>.

- **Framework**. [En línea] 19 de enero de 2009. [Citado el: 22 de enero de 2009.] <http://es.wikipedia.org/wiki/Framework>.
- **Entornos gráficos 2D**. *Kioskea.net*. [En línea] <http://es.kioskea.net/faq/sujet-499-entornos-graficos-2d#xfce>.
- **Entornos de escritorio**. *Manual de FreeBSD*. [En línea] http://www.freebsd.org/doc/es_ES.ISO8859-1/books/handbook/x11-wm.html.
- **Ejemplos java y C/linux**. [En línea] [Citado el: 2009 de enero de 23.] http://www.chuidiang.com/java/novatos/editor/leer_escribir_fichero.php.
- **Beyond Linux From Scratch** - Versión 6.0. [En línea] 9 de abril de 2005. [Citado el: 2009 de enero de 24.] <http://www.escomposlinux.org/lfs-es/blfs-es-6.0/general/glib2.html>.
- **¿Qué es KDE?** *KDE*. [En línea] 2009. <http://docs.kde.org/stable/es/kdebase-runtime/khelpcenter/what-is-kde.html>.
- **app-admin/augeas**. *Gentoo-Portage*. [En línea] <http://gentoo-portage.zugaina.org/app-admin/augeas/ChangeLog>.
- **Linux Kernel Development** (April 2008). *The Linux Foundation*. [En línea] <http://www.linuxfoundation.org/publications/linuxkerneldevelopment.php>.
- **The history of the Linux user count**. *Linux Counter Summary Report*. [En línea] <http://counter.li.org/reports/short.php>.
- **Bidirectional Programming Languages**. *Social Web*. [En línea] <http://www.socialweb.net/Events/90717.lasso>.
- **A bidirectional programming language for ad-hoc, textual data**. . *Boomerang*. [En línea] <http://alliance.seas.upenn.edu/~harmony/>.
- **harmony-header**. [En línea] <http://www.seas.upenn.edu/~harmony/old/>.

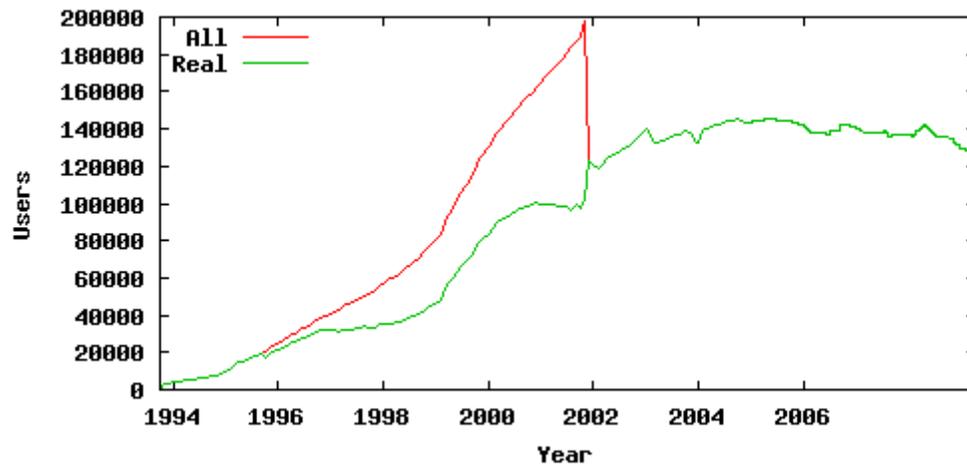
Anexos

Anexo 1. Entornos de escritorios más utilizados.

Ver Resultados de Encuesta: ¿Qué tipos de entornos de escritorio habéis usado?			
KDE		464	89,40%
GNOME		317	61,08%
XFCE		71	13,68%
Enlightenment		42	8,09%
Fluxbox		30	5,78%
WindowMaker		52	10,02%
FVWM		25	4,82%
ROX		5	0,96%
Otros...		13	2,50%
Línea de comandos		71	13,68%

Encuesta de Elección Múltiple. Votantes: 519.

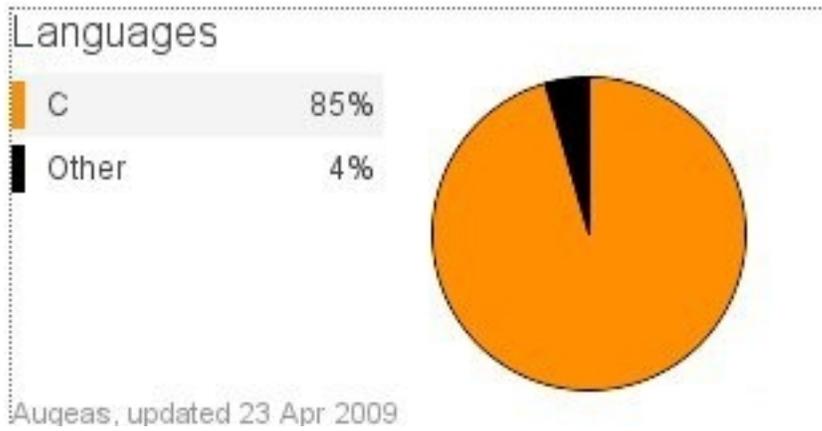
Anexo 2. Listado de usuarios de GNU/Linux por años.



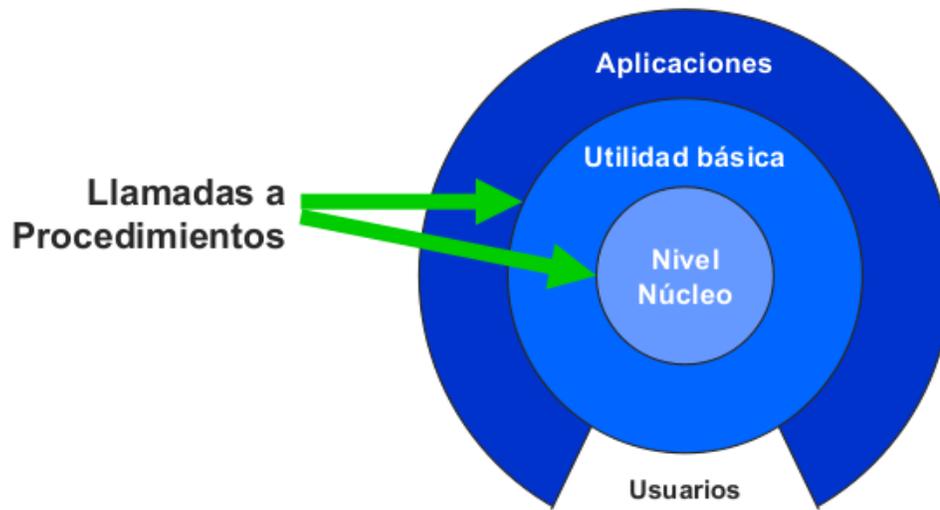
Anexo 3. Listado de compañías que patrocinan a desarrolladores del kernel de Linux.

Company Name	# of Changes	% of Total
None	11,594	13.9%
Unknown	10,803	12.9%
Red Hat	9,351	11.2%
Novell	7,385	8.9%
IBM	6,952	8.3%
Intel	3,388	4.1%
Linux Foundation	2,160	2.6%
Consultant	2,055	2.5%
SGI	1,649	2.0%
MIPS Technologies	1,341	1.6%
Oracle	1,122	1.3%
MontaVista	1,010	1.2%
Google	965	1.1%
Linutronix	817	1.0%
HP	765	0.9%
NetApp	764	0.9%
SWsoft	762	0.9%
Renesas Technology	759	0.9%
Freescale	730	0.9%
Astaro	715	0.9%
Academia	656	0.8%
Cisco	442	0.5%
Simtec	437	0.5%
Linux Networx	434	0.5%
QLogic	398	0.5%
Fujitsu	389	0.5%
Broadcom	385	0.5%
Analog Devices	358	0.4%
Mandriva	329	0.4%
Mellanox	294	0.4%
Snapgear	285	0.3%

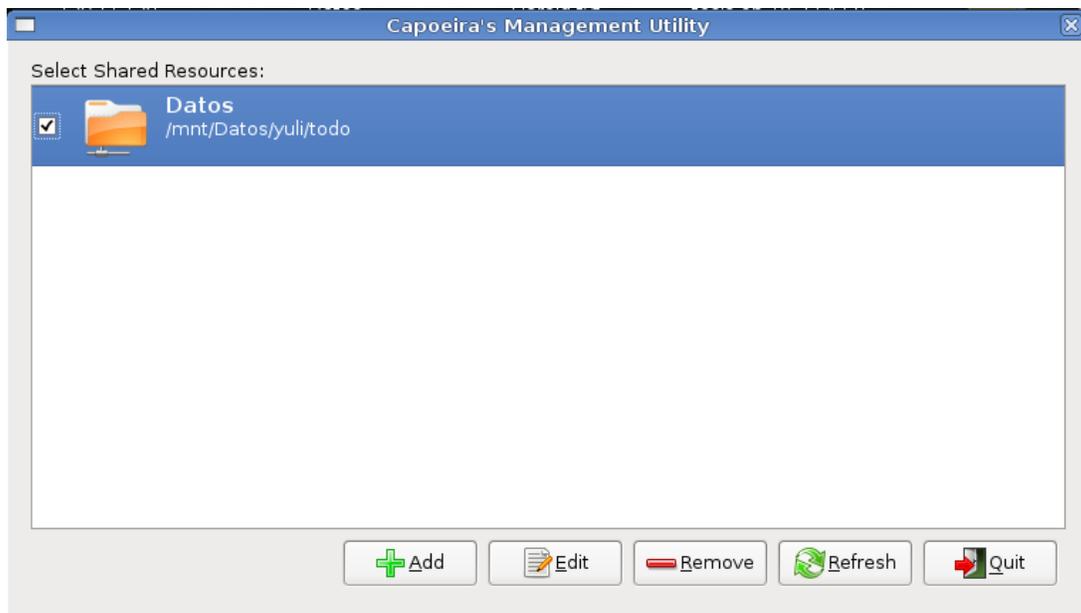
Anexo 4. Lenguajes de programación de Augeas



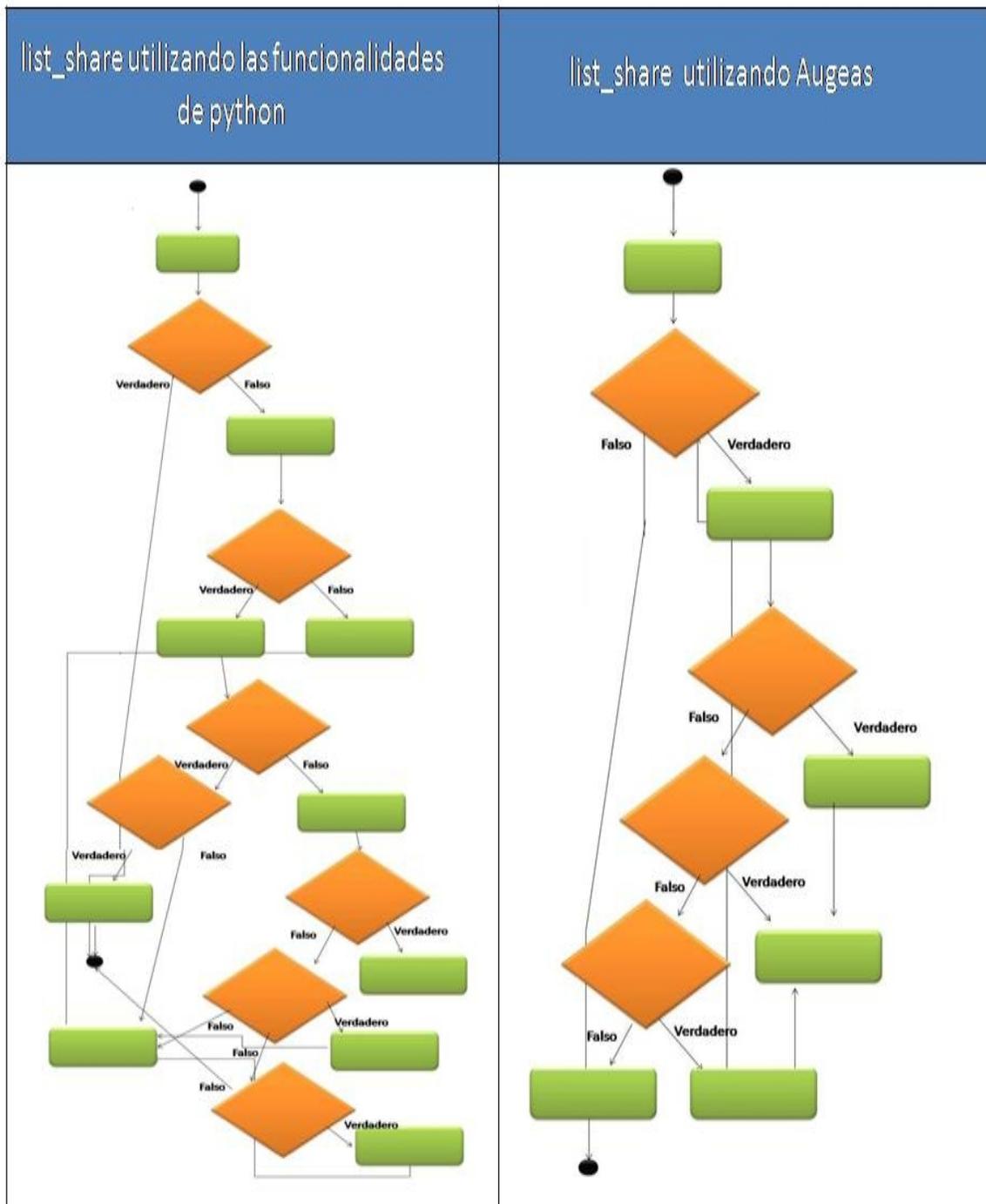
Anexo 5. Arquitectura en capas



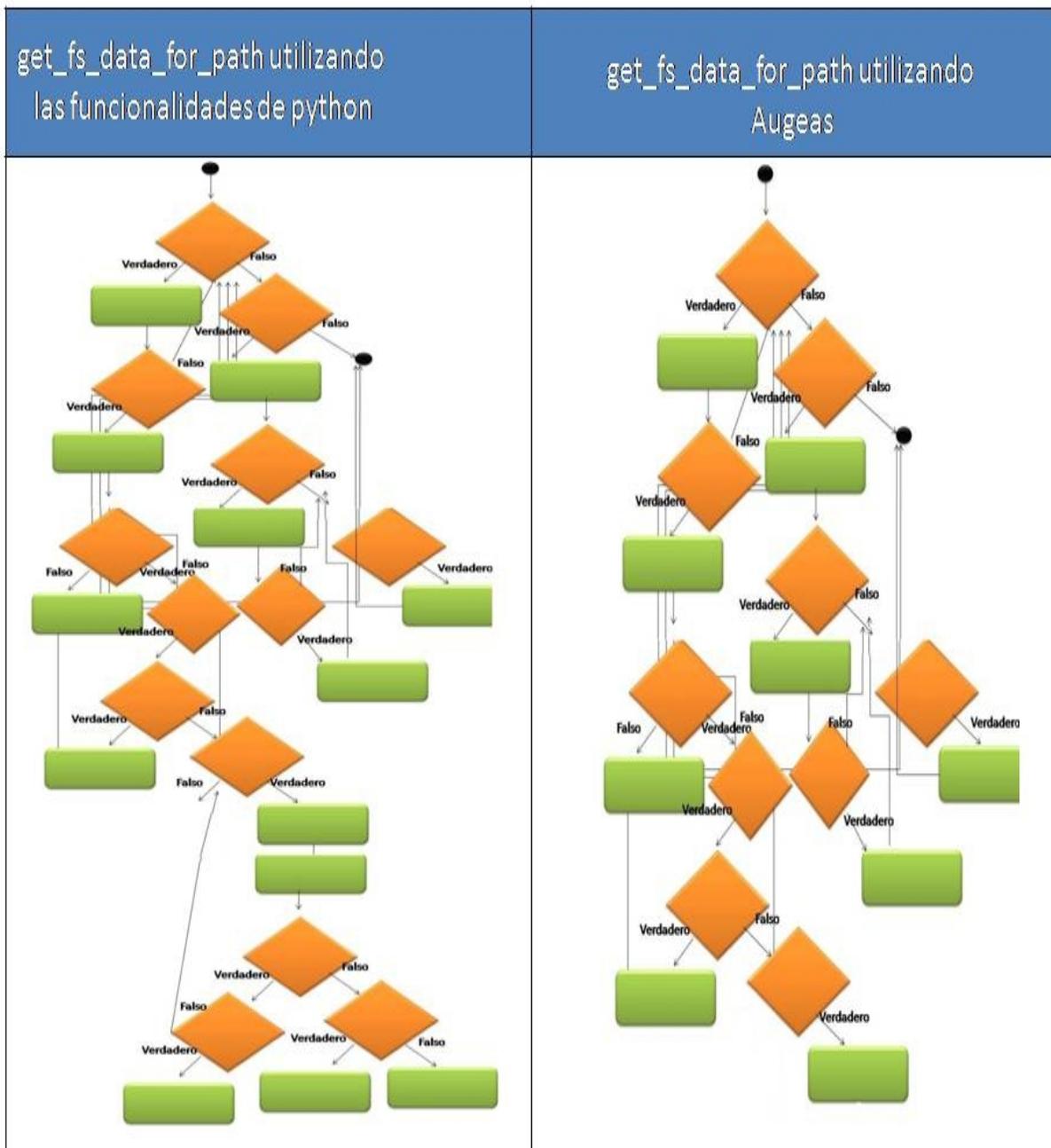
Anexo 6. Capoeira



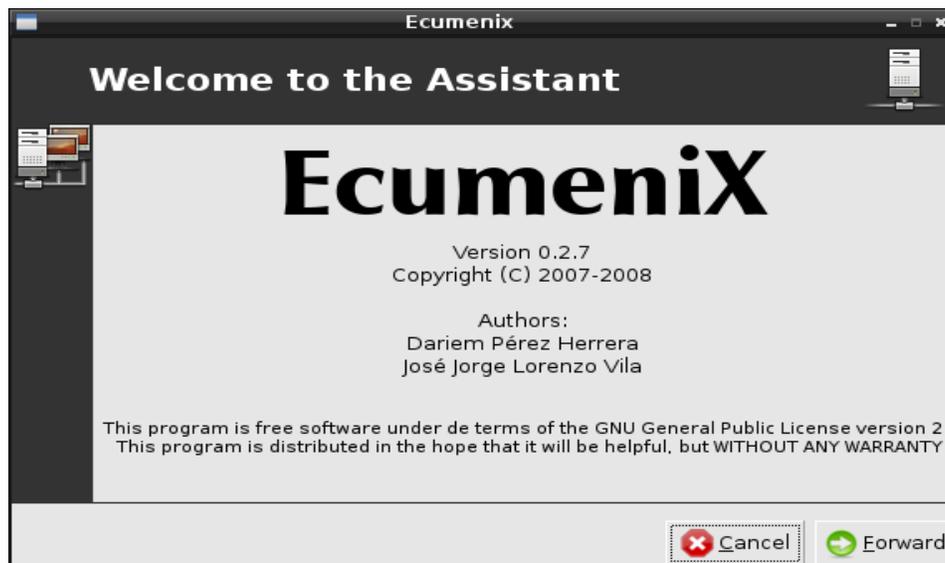
Anexo 7. Comparación del método list_share (sin utilizar Augeas) / list_share (utilizando Augeas)



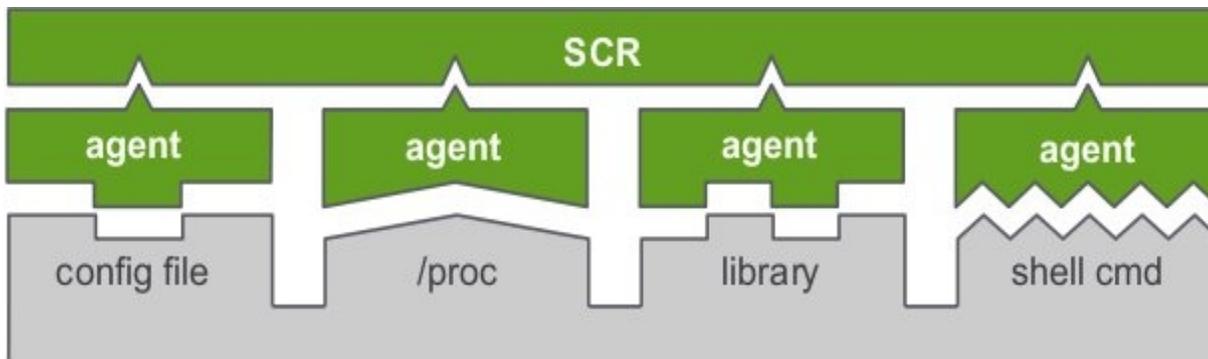
Anexo 8. Comparación del método `get_fs_data_for_path` (sin utilizar Augeas) / `get_fs_data_for_path` (utilizando Augeas)



Anexo 9. EcumeniX

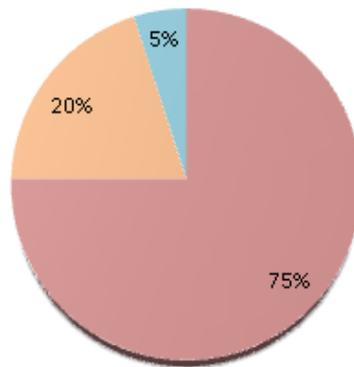


Anexo 10. SCR



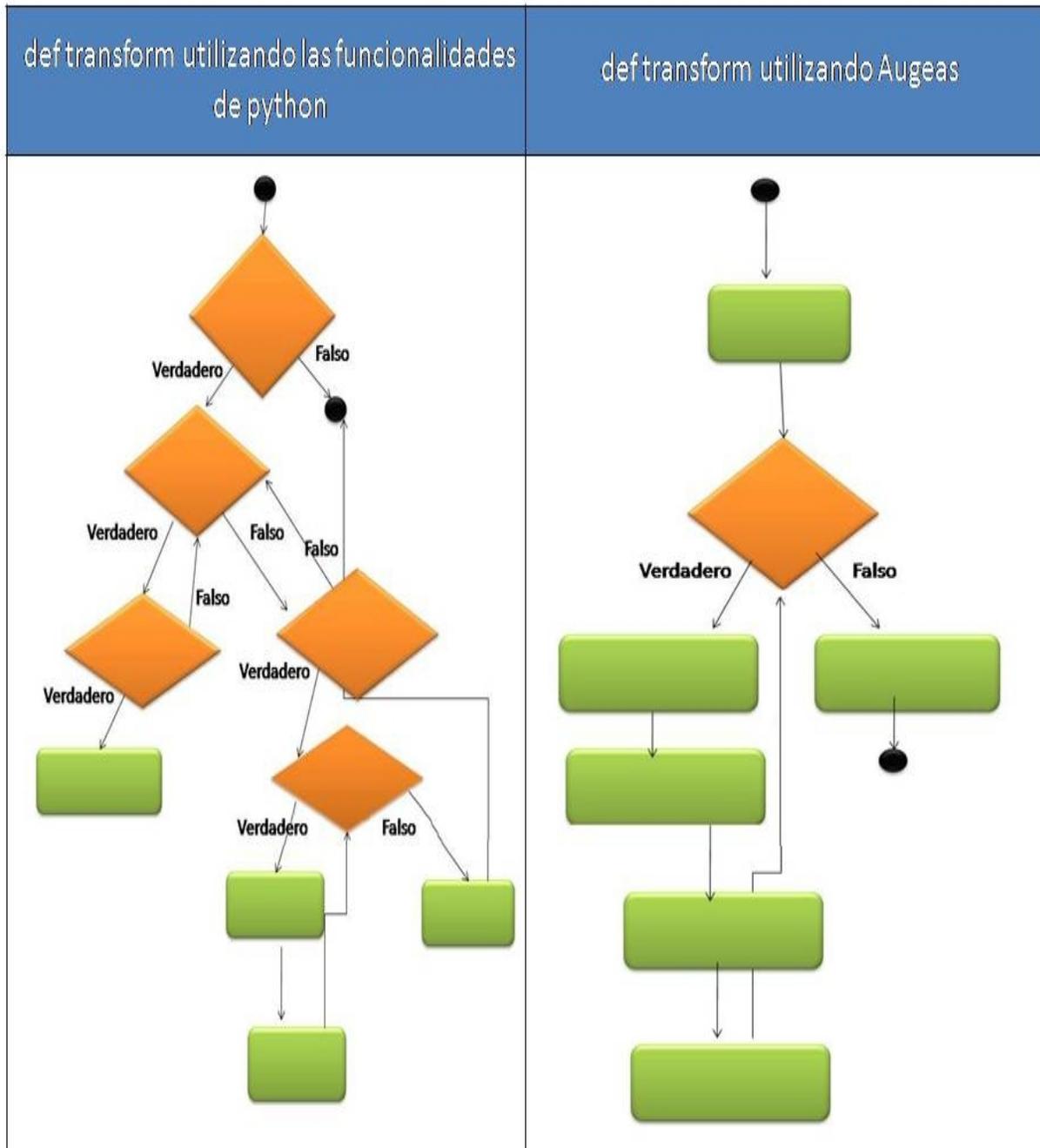
Anexo 11. Resultados de la encuesta

Tecnologías que utilizan los desarrolladores para la gestión de archivos

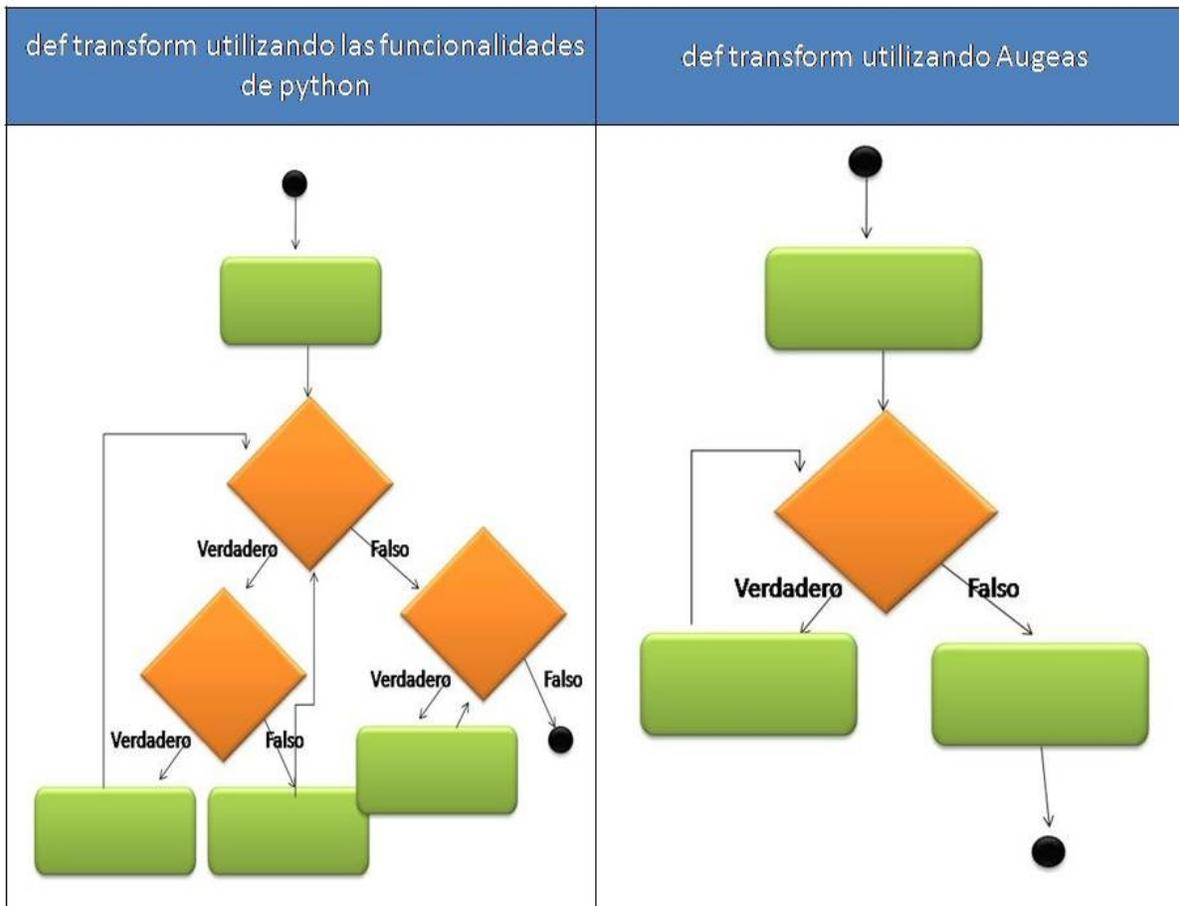


- : Librerías y funciones que brinda el lenguaje de programación.
- : Módulos creados para el manejo de ficheros.
- : Ambas

Anexo 12. Comparación del método deftransform (sin utilizar Augeas) / deftransform (utilizando Augeas) utilizado en el fichero hostsparser.py



Anexo 13. Comparación del método deftransform (sin utilizar Augeas) / deftransform (utilizando Augeas) utilizado en el fichero ntparser.py



Anexo 14. Glosario de Términos

Ad hoc: Sistemas desarrollados para alguna aplicación determinada, que utilizan un sistema de manejo de archivos propio y por ende un sistema de administración de datos propio.

API: Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Backend: El software que hace el trabajo duro, como por ejemplo el "modelo" en el modelo vista controlador.

Bcfg2: Es una herramienta de administración para la configuración.

Biblioteca: Término informático para referirse a las bibliotecas de vínculos dinámicos, conocidas también como librerías.

C: Es un lenguaje de programación creado en 1972 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL. Es orientado a la implementación de Sistemas Operativos, concretamente Unix

Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix

Cfengine: Es una herramienta de configuración del sistema basada en políticas de seguridad.

Consola: Programa que tiene como objetivo principal, la interacción del usuario con el sistema operativo a base de líneas de comandos. Las consolas más usadas por los usuarios son las consolas virtuales, que son las que se pueden ver desde su entorno gráfico, por ejemplo: Gnome-terminal, Kterm y Yaquake.

Consumo de recursos: Término informático que se refiere a la forma en que el software interactúa con los recursos de hardware de un ordenador.

DBus: Sistema de comunicación interprocesos de Linux.

Distribuciones o Distros: Una distribución GNU/Linux (llamadas también distribuciones Linux) es una variante de ese sistema operativo que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones hogareñas, empresariales y para servidores. Pueden ser exclusivamente de software libre o también incorporar aplicaciones o controladores privativos.

Gentoo: Gentoo Linux es una distribución GNU/Linux orientada a usuarios con cierta experiencia en este sistema operativo, fue fundada por Daniel Robbins, basada en la inactiva distribución llamada Enoch Linux.

GNU/Linux: GNU/Linux es, a simple vista, un Sistema Operativo. Es una implementación de libre distribución UNIX para computadoras personales, servidores, y estaciones de trabajo. Fue desarrollado para el i386 y ahora soporta los procesadores i486, Pentium, Pentium Pro y Pentium II, así como los clones AMD y Cyrix. También soporta máquinas basadas en SPARC, DEC Alpha, PowerPC/PowerMac, y Mac/Amiga Motorola 680x0.

Como sistema operativo, GNU/Linux es muy eficiente y tiene un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador; en las plataformas Intel corre en modo protegido; protege la memoria para que un programa no pueda hacer caer al resto del sistema; carga sólo las partes de un programa que se usan; comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria; usa un sistema de memoria virtual por páginas; utiliza toda la memoria libre para cache; permite usar bibliotecas enlazadas tanto estática como dinámicamente; se distribuye con código fuente; usa hasta 64 consolas virtuales; tiene un sistema de archivos avanzado pero puede usar los de los otros sistemas; y soporta redes tanto en TCP/IP como en otros protocolos.

GTK+: Bibliotecas para desarrollar interfaces gráficas de usuarios.

Haskell: Es un lenguaje de programación puramente funcional de propósito general y fuertemente tipificado. Su nombre proviene del lógico Haskell Curry.

Interfaces (API): API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Interfaz gráfica (GUI): En el contexto del proceso de interacción persona-ordenador, la interfaz gráfica de usuario, es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un

sistema informático.

Java: Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

Licencia libre: Copyleft o copia permitida comprende a un grupo de derechos de propiedad intelectual caracterizados por eliminar las restricciones de distribución o modificación de las que adolece el copyright, con la condición de que el trabajo derivado se mantenga con el mismo régimen de propiedad intelectual que el original.

Pueden protegerse una gran diversidad de obras, tales como programas informáticos, arte, cultura y ciencia, es decir prácticamente casi cualquier tipo de producción creativa.

Sus partidarios la proponen como alternativa a las restricciones que imponen las normas planteadas en los derechos de autor, a la hora de hacer, modificar y distribuir copias de una obra determinada. Se pretende garantizar así una mayor libertad para que cada receptor de una copia, o una versión derivada de un trabajo, pueda, a su vez, usar, modificar y redistribuir tanto el propio trabajo como las versiones derivadas del mismo. Así, y en un entorno no legal, puede considerarse como opuesto al copyright o derechos de autor tradicionales.

LINUX: Núcleo presente en los sistemas GNU/Linux.

MFC: Microsoft Foundation Class Library es una biblioteca que envuelve porciones de la API de Windows en clases de C++, incluyendo la funcionalidad que las habilita para usar la plataforma de la aplicación por defecto.

ML: Es un lenguaje de programación de propósito general de la familia de los lenguajes de programación funcional desarrollado.

NaturalDocs: Es un generador de documentación multi-lenguaje. Fue escrito en Perl y está disponible como software libre bajo la licencia GNU General Public License.

.NET: Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

Ocaml: El lenguaje Objective CAML, también llamado Ocaml o O'Caml, es un lenguaje de programación avanzado de la familia de los lenguajes ML, desarrollado y distribuido por el

INRIA en Francia. Ocaml admite los paradigmas de programación imperativa, programación funcional y programación orientada a objetos.

Perl: Es un lenguaje de programación diseñado por Larry Wall en 1987.

PolicyKit: Es una aplicación de herramientas de desarrollo para el control de todo el sistema de privilegios en sistemas operativos del tipo UNIX.

Puntero: Es una variable cuyo valor es una dirección de memoria.

Puppet: Es una herramienta de administración para la configuración de los sistemas Unix.

Python: Es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991.

Rendimiento: Un término usado en la informática para dar medida de cuán eficaz puede ser un software en el uso de los recursos de hardware.

Ruby: Es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995. Combina una sintaxis inspirada en Python, Perl con características de programación orientada a objetos similares a Smalltalk.

Sistema operativo: Un sistema operativo es un software de sistema, es decir, un conjunto de programas de computadora destinado a permitir una administración eficaz de sus recursos. Comienza a trabajar cuando se enciende el computador, y gestiona el hardware de la máquina desde los niveles más básicos, permitiendo también la interacción con el usuario.

Terminal: Término derivado del inglés que significa consola.

UNIX: Unix (registrado oficialmente como UNIX) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.