Universidad de las Ciencias Informáticas

Facultad 1

Título: Módulo para la gestión remota de los servicios que componen al sistema Smart Keeper v2.0.

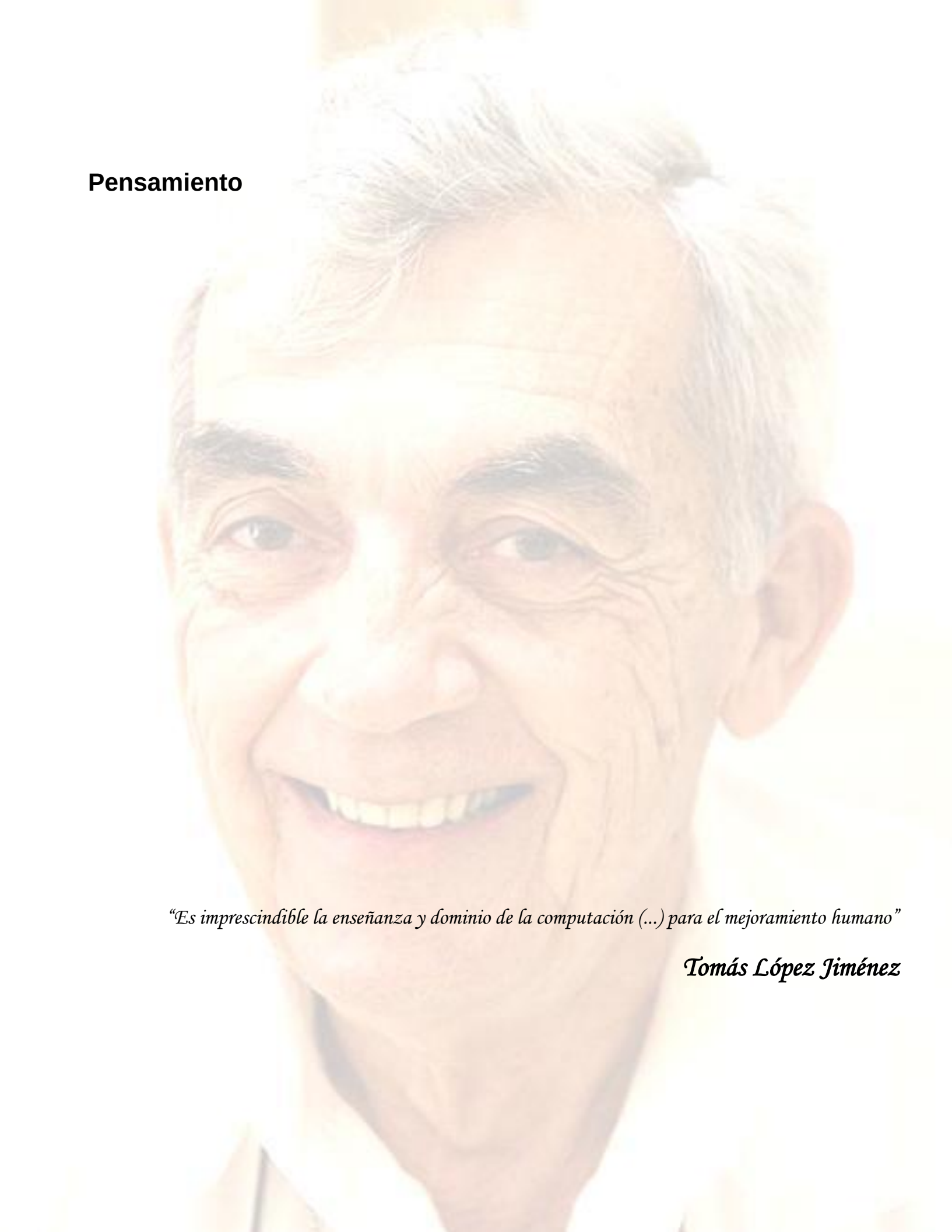
Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores: Ruben Reynaldo Bonachea
Goar Espinosa Marrero

Tutores: Ing. Daileny Hernández Barreiro
Ing. Ariagna González Landeiro

La Habana, Cuba
Junio, 2012
"Año 54 de la Revolución"

Pensamiento



"Es imprescindible la enseñanza y dominio de la computación (...) para el mejoramiento humano"

Tomás López Jiménez

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

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

Ruben Reynaldo Bonachea	Goar Espinosa Marrero
Ing. Daileny Hernández Barreiro	Ing. Ariagna González Landeiro

Agradecimientos

A mis padres y hermana por todo el apoyo, el cariño y la dedicación que me han dado.

A mis abuelos.

A Roberto y a Teresa.

A mis primos Robinson, Wendy, Chacha, Heribertico, Nailat y a todos los que no mencione también.

A mis tías Belkis, Dayami, Maribel.

A mis tíos Raúl, Reine, Omar, Osmani, Juan, Clemente, Fito.

A la gente del Barrio, son muchos para poner sus nombres.

A mis amigos de aquí de la UCI Yasmani, Ernesto, Carlos, Johan, Alejandro, Lara, Manuel y Jorge.

A todos mis profesores.

A mi compañero de Tesis.

A todos los que mencioné y a los que de una forma u otra me han ayudado a llegar hasta aquí, mis más sinceros agradecimientos.

Ruben Reynaldo Bonachea

A mis compañeros que junto nos hemos formado en esta magnífica universidad y defendimos la facultad y el Software libre. A los que estuvieron junto a mí cumpliendo misión internacionalista.

A mis compañeros de la FEU durante estos 5 años en la universidad, con los que me divertí, trabajé y consagré.

Agradecimientos

A mis hermanas y hermanos de la aldea, aquellos con los que compartí mi infancia y momentos muy importantes de mi vida y siempre me han tendido una mano de ayuda.

En especial a mi compañero de tesis, por las horas que compartimos en el proyecto para cumplir con nuestro trabajo de diploma.

A todos los que estuvieron a mi lado y dejaron en mi camino un momento de felicidad y un grano de sabiduría.

… los voy a extrañar.

Goar Espinosa Marrero

Ambos agradecemos:

A nuestros compañeros de Proyecto Eddy, Juan Carlos, Eduardo, Yuri, Luis, Enier y Juan Manuel.

A nuestras tutoras Ariagna y Daileny por su interés, dedicación y entrega, esta tesis también es de ustedes.

Es largo el camino que se anda durante 5 largos años, pero el mismo se hace corto, si te sientes apoyado por las personas que realmente te quieren. A ellos dedico este trabajo que lleva todo mi esfuerzo y dedicación. Ellos son mis padres Ruben e Isabel que siempre me han apoyado en todo y han estado ahí en cada momento. A mi hermanita Elizabeth, a mi abuela María, a las personas que me han aceptado como un hijo Tere y Roberto. A mis primos que me apoyan a cada momento Robinson, Osbaldito, Wendy, Chacha, Heribertico, Nailat. A mis tíos y tías.

Ruben Reynaldo Bonachea

A mis padres Clara y Goar que me educaron con un profundo sentido del deber y del trabajo, a quienes debo mis ganas de superarme, de ser alguien en la vida, de luchar por lo que quiero y ser útil a la sociedad. Son mi ejemplo y mi apoyo.

… gracias por traerme a este mundo y darme un espacio en sus corazones.

A mis hermanas, en especial a la más pequeñita (Yuli), nadie me ha cuidado como tú mientras nuestra madre ha estado lejos.

A mis abuelos Clara y Rogelio por tenerme siempre presente y quererme tanto. A toda mi familia que tanto apoyo me ha dado durante toda la carrera para poder lograr mi sueño de graduarme a pesar de la distancia.

A mi novia, una mujer extraordinaria, a quien amo profundamente, me has ayudado a ser mejor persona, a cumplir mis sueños, a conocer el verdadero significado de la vida.

…gracias por tu cariño.

Goar Espinosa Marrero

Una buena administración y configuración de un sistema es de vital importancia para su correcto funcionamiento. El filtro de contenido web Smart Keeper desarrollado en la Universidad de las Ciencias Informáticas (UCI) no poseía ningún mecanismo que le permitiera gestionar cada uno de los servicios que lo componen. Esto traía como consecuencia la pérdida de tiempo a la hora de configurar un determinado servicio o conocer el estado del sistema.

El objetivo de este trabajo fue el desarrollo de un módulo que permitiera de forma remota, administrar y configurar los servicios que componen al sistema Smart Keeper. Para esto se realizó un estudio detallado de las herramientas más apropiadas para el desarrollo del módulo, así como a los protocolos y herramientas utilizadas para realizar conexiones remotas.

Se definió de forma general el funcionamiento del módulo, se diseñó y se implementó. Por último, se evaluó el trabajo realizado y se integró a Smart Keeper. De esta forma los administradores, actualmente, pueden gestionar todos los servicios de este sistema desde su misma interfaz de administración web. Donde todas las acciones sobre los servidores se realizan de forma rápida y segura, utilizando el protocolo SSH2. En cada servicio se brinda la posibilidad de iniciarlo, detenerlo, reiniciarlo, restaurarlo y cambiar sus parámetros de configuración. Además, se permite crear y leer copias de respaldo de los ficheros de configuración así como monitorizar el correcto funcionamiento del sistema.

Palabras clave: administrar, configurar, módulo, remoto, servicios, Smart Keeper.

A good management and configuration of a system is vital to its correct performance. The web content filter Smart Keeper developed in the University of the Informatics Sciences (UCI) did not have any mechanism to manage each one of the services that compose it. This brought as a consequence the waste of time to configure a specific service or knowing the system status.

The objective of this work was the development of a module to allow the remote management and configuration of the services that compose Smart Keeper system. To do it, was made a detailed study of the most appropriated tools for the module's development, likewise to the protocols and tools used to make remote connections.

In a general way was defined the module's performance, it was designed and implemented. Finally, the work done was evaluated and integrated to Smart Keeper. By this way the administrators, currently, can manage all the services of this system form the same web management interface. Where all the actions over the servers are made in a fast and secure way, using SSH2 protocol. Each service can be started, stopped, restarted, restored and its configuration parameters changed. Also allows creating and reading backup copies of the configuration files such as monitoring the correct performance of the system.

***Keywords**: configure, manage, module, remote, services, Smart Keeper.*

Índice de figuras

Índice de tablas

Introducción

En Internet cada vez es más amplia y diversa la gama de contenidos que a través de ella transita. Los internautas a cada minuto acceden y comparten recursos informativos, recreativos, científicos y educativos. De esta misma forma existen materiales que pueden resultar nocivos, ilícitos e incluso ilegales en muchos países dependiendo de legislaciones que tengan implantadas. Debido a esto se hace de suma importancia regular y monitorizar el acceso de los usuarios a la red de redes. Una solución técnica a esta situación es el filtrado de contenidos.

Los sistemas de filtrado de contenidos se utilizan para regular el acceso de los usuarios a Internet en empresas, centros de enseñanza y hogares, definiendo políticas acordes con sus necesidades. Existen hoy en día en el mercado diversas técnicas de selección y filtrado de contenidos [1], ejemplos de estas son:

- Herramientas de control de acceso y monitorización para los ordenadores.
- Herramientas de control de acceso en el proveedor de servicios de Internet (ISP).

Estas herramientas funcionan de distintas maneras para evitar el acceso a los contenidos inadecuados:

- Bloqueando direcciones de páginas web que contengan dichos contenidos.
- Controlando horas de acceso.
- Permitiendo establecer una lista propia de direcciones aceptadas o negadas.
- Asignando diferentes perfiles en diferentes días y horas (trabajo, tiempo libre, etc.).
- Permitiendo regular qué servicios se pueden utilizar en cada momento y por cada usuario (correo, *chat*, etc.).

Smart Keeper[1] es un filtro de contenidos, desarrollado en la Universidad de las Ciencias Informáticas (UCI). Surgió a solicitud de la Oficina de Seguridad para las Redes Informáticas (OSRI), en el año 2005, con el nombre FILPACON[2] [2]. Este sistema está dirigido a regular el acceso a Internet en empresas e instituciones. Está basado en el uso e integración de varios servicios[3]: un servidor *proxy* Squid, un servidor

[1] Traducción de los autores: Guardián Inteligente.

[2] Acrónimo de Filtrado de Paquetes por Contenido.

[3] Aplicaciones que ofrecen prestaciones del tipo cliente-servidor.

web Apache, un servidor de base de datos PostgreSQL, un servidor *Internet Content Adaptation Protocol*[4] (ICAP) y otros componentes libres. Estos poseen diversos ficheros de configuración y de registros, además pueden ser iniciados, detenidos o reiniciados. Esto hace que Smart Keeper tenga gran cantidad de archivos de configuración y genere un importante número de registros por cada uno de los servicios que lo componen. El sistema permite además ser instalado de forma distribuida como se puede apreciar en la figura 1, del Anexo A [3].

Este *software* cuenta además con una interfaz web que posibilita a los administradores gestionar y monitorizar el acceso de los usuarios a Internet [4]; pero no incluye ninguna funcionalidad que permita de forma remota y centralizada el manejo de los archivos de configuración y la administración de los servicios que lo conforman. Esta carencia es un obstáculo para la facilidad de administración del sistema. Actualmente, para que un administrador pueda realizar sus tareas en cada uno de los servicios, cuenta con las siguientes opciones:

- Mediante un terminal conectarse a cada uno de los servidores donde se encuentran instalados dichos servicios y hacer las configuraciones pertinentes.
- En cada uno de los servidores, tener instaladas herramientas que permitan la administración remota de los servicios.
- Administrar los servicios de forma local.

La primera opción es difícil e incómoda porque trae consigo la necesidad de conocer numerosos comandos, estructuras de los directorios, ficheros e igualmente dominar el trabajo mediante la terminal. La segunda vía provoca que se necesite más tiempo para administrar los servicios, pues se tienen que realizar las tareas independientemente en cada uno de ellos. Además, es necesario tener instaladas en los servidores herramientas específicas para cada servicio. Estas opciones añaden además problemas de seguridad, pues no todas las herramientas para administrar y configurar servicios en servidores remotos proveen una forma segura para la transferencia de información. La última opción es la más ineficiente y por tanto menos viable pues si se ubican los servidores en locales diferentes, ocasiona que el administrador necesite trasladarse hacia dichos locales en caso de necesitar realizar alguna configuración.

La situación anterior permite identificar el siguiente **problema de investigación**: ¿Cómo mejorar el

[4] Traducción de los autores: Protocolo de Adaptación de Contenidos de Internet.

proceso de administración y configuración de los servicios que componen al sistema Smart Keeper?

El **objeto de estudio** del presente Trabajo de Diploma está definido por el proceso de administración y configuración de servicios informáticos.

El **campo de acción** está enmarcado en el proceso de administración y configuración remota de servicios informáticos.

Como **idea a defender** se establece: El desarrollo de un módulo para la gestión remota de los servicios que componen a Smart Keeper, facilitará el trabajo a los administradores del sistema.

El **objetivo general** se define como: Desarrollar un módulo que permita, de forma remota, administrar y configurar los servicios que componen al sistema Smart Keeper.

Del objetivo general enunciado se desglosan los siguientes **objetivos específicos**:

- Sistematizar acerca de aplicaciones que permitan administrar servicios de manera remota.
- Identificar las características que contendrá la solución propuesta para administrar los servicios de manera remota en Smart Keeper.
- Implementar las funcionalidades del módulo.
- Validar las funcionalidades del módulo.

Para lograr los objetivos enunciados se plantean las siguientes **tareas**:

- Caracterización de aplicaciones que permitan administrar servicios de manera remota.
- Conceptualización de los elementos necesarios para comprender el ámbito del problema a resolver y la solución que se proponga.
- Análisis de protocolos y herramientas para realizar conexiones remotas en GNU/Linux.
- Selección de las herramientas, tecnologías y metodología de desarrollo a emplear para la concepción de la solución propuesta.
- Análisis y diseño de la solución propuesta en dependencia de la metodología que se determine utilizar para el desarrollo.

- Implementación de la solución en forma de módulo para ser integrada a la interfaz de administración de Smart Keeper.
- Integración del módulo al sistema Smart Keeper.
- Validación de la solución obtenida a través de pruebas funcionales y de integración.

Para el cumplimiento de las tareas propuestas se utilizarán los siguientes **métodos** teóricos:

- El Analítico-Sintético se aplicará para entender las aplicaciones que se utilizan para configurar y administrar servicios en GNU/Linux, con el objetivo de formular conclusiones a través de la síntesis de los conocimientos y resultados obtenidos.
- El Histórico-Lógico permitirá una mayor comprensión del estado actual de las aplicaciones que administran y configuran servicios en GNU/Linux, a partir del análisis de su evolución y las etapas principales por las que han transitado.

El presente Trabajo está estructurado en cuatro capítulos tal como se describe a continuación:

El **primer capítulo** se centra en la investigación del tema relacionado con el objeto de estudio. Se analiza la existencia de soluciones y la posibilidad de que se integren a Smart Keeper. Se abordan temas referentes a los diferentes servicios que componen al sistema y a las principales aplicaciones que permiten administrar servicios de manera remota. Además, se seleccionan las principales tecnologías para el desarrollo del módulo.

En el **segundo capítulo** se tratan los principales aspectos de los flujos de trabajo Modelamiento del Negocio y Requisitos. Se conceptualiza el entorno mediante un modelo de dominio, se especifican los requisitos funcionales y no funcionales que cumple la solución. Se presentan los diagramas de Casos de Uso del Sistema, se describen los actores y se detallan los Casos de Uso del Sistema.

En el **tercer capítulo** se exponen los principales artefactos generados durante los flujos de trabajo Análisis y Diseño. Se define la estructura del diseño de la aplicación desarrollada, y para ello se representa el diagrama de clases del diseño, el diagrama de clases persistentes y el modelo Entidad-Relación, así como el modelo de datos del componente.

En el **cuarto capítulo** se exponen los principales artefactos generados durante los flujos de trabajo Implementación y Prueba.

Capítulo 1

Administración y configuración de Servicios

Introducción

En la rama informática a una aplicación que brinda prestaciones del tipo Cliente-Servidor se le llama servicio [5]. Actualmente existen gran cantidad de servicios, una de sus características es que pueden ser gestionados (configurar, iniciar, detener, restaurar, reiniciar) a través de una terminal o usando aplicaciones gráficas que poseen los distintos sistemas operativos. Usar estas aplicaciones gráficas permite que desde una misma interfaz se puedan realizar un gran número de acciones sobre los servicios, como son: cambiar niveles de ejecución, indicar si el servicio arranca o se detiene, incluso iniciarlos o detenerlos en ese momento. Los servicios pueden ser gestionados tanto de forma local como remota utilizando disímiles herramientas.

En este capítulo se hará un estudio de las aplicaciones existentes para administrar servicios, así como las diferentes herramientas y protocolos usados para realizar conexiones remotas. Además, se seleccionarán las tecnologías y herramientas adecuadas para dar solución a la situación problemática.

1.1 Estado del arte

A nivel internacional existen varios filtros de contenidos como son: Optenet[5], PureSight[6] y Censornet[7]. En el caso de los dos primeros no cuentan con una herramienta que permita la configuración de servicios y pueda ser integrada a Smart Keeper debido a que son sistemas propietarios.

Censornet, en su versión libre, la cual se desarrolló hasta el 2005, tiene integrada la opción de cambiar algunas variables en los ficheros de configuración de los servicios que lo componen y además ofrece la funcionalidad de reiniciar cada uno de ellos. La limitante que presenta y por lo que se descarta dicha

[5] Página oficial. <http://www.optenet.com>.

[6] Página oficial. <http://www.icognito.com>.

[7] Página oficial. <http://www.censornet.com>.

solución es se deben instalar todos sus componentes de forma centralizada ya que las acciones sobre estos se realizan localmente.

Además de las herramientas integradas con los sistemas de filtrado existen otras que su único propósito es la configuración de servicios ya sea de manera local o remota. A continuación se describen algunas con sus principales características.

Webmin: Es una herramienta genérica de administración. Posee una interfaz basada en web que permite la administración de Sistemas Unix. El objetivo de Webmin es poner de forma accesible la configuración de la mayoría de los programas/servicios que se usan normalmente en Unix, de modo que todo se logra mediante formularios web, pero a su vez, se pueden editar los archivos de configuración en modo editor de texto. Dentro de sus principales características se encuentra que es una aplicación modular a la cual se le pueden instalar nuevos módulos [6].

Linuxconf*:* Es una herramienta genérica de administración, la cual permite configurar y controlar varios aspectos del sistema. Posee 3 tipos de interfaces de usuario, una de menús textuales, una web y una de interfaz de línea de comandos. Puede utilizarse en prácticamente cualquier distribución GNU/Linux [7].

YAST (*Yet Another System Tool*): Es la herramienta de instalación y configuración de openSUSE, SUSE Linux Enterprise y otras distribuciones SUSE Linux. Es popular por su facilidad de uso y atractiva interfaz gráfica, así como la capacidad de configurar su sistema rápidamente durante y luego de la instalación. Se utiliza para configurar dispositivos, el entorno de red, los servicios del sistema y afinar su configuración de seguridad [7].

Zentyal: Es la alternativa en código abierto a *Windows Small Business Server*[8], basado en la distribución Ubuntu. Zentyal permite administrar de forma sencilla y a través de una única plataforma todos los servicios de una red informática, tales como: el acceso a Internet, la seguridad de la red, la compartición de recursos, la infraestructura de la red o las comunicaciones. Una de sus características más importantes es que todas sus funcionalidades, construidas a partir de una serie de aplicaciones en principio independientes, están estrechamente integradas entre sí, automatizando la mayoría de las tareas y ahorrando tiempo en la administración de sistemas [8].

[8] Suite integrada de servidor desarrollada por Microsoft, diseñada para el funcionamiento de la infraestructura de una red.

Las aplicaciones mencionadas anteriormente no constituyen una solución aplicable al filtro de contenidos web Smart Keeper, debido a que ninguna permite la configuración de servicios de forma remota sin necesidad de tener que instalar la respectiva aplicación donde se encuentre el servicio a configurar. Además, se suman otros inconvenientes. Toda la administración del sistema Smart Keeper se realiza vía web mediante su interfaz de administración. Por lo que se requiere que todas las funcionalidades que sean añadidas al sistema puedan ser integradas a su interfaz.

Dado que es imposible aprovechar alguna solución existente, es necesario desarrollar una solución a la medida que se integre con la interfaz de Smart Keeper.

1.2 Servicios en GNU/Linux

Los servicios en GNU/Linux se pueden encontrar principalmente en el directorio ***/etc/init.d*****,** donde están los *scripts*[9] de cada uno ellos. Además, existe una serie de directorios ***/etc/rc.N*** donde N es el *runlevel* (nivel de ejecución), que contienen enlaces simbólicos[10] a los *scripts* de /etc/init.d. Existen otros servicios, normalmente de red, que no se ejecutan como demonios[11], si no bajo demanda, estos servicios se controlan desde un archivo de texto en ***/etc/inetd.conf***.

Los servicios en GNU/Linux se engloban dentro de las siguientes categorías [9]:

- **Aplicaciones:** El servidor dispone la ejecución de aplicaciones y los clientes únicamente pueden interactuar con ellas.
- **Ficheros:** Proporcionan un espacio común y accesible desde cualquier punto de la red de donde se puede almacenar/recuperar ficheros.
- **Base de datos:** Centralizan datos que se van a consultar o producir por parte de las aplicaciones del sistema en red (o bien de otros servicios).
- **Impresión:** Gestionan los trabajos de impresión que se les envíen desde cualquier punto de la red.

[9] Archivo de órdenes para interactuar con el sistema operativo, también conocidos como shellscripts.

[10] Acceso a un directorio o fichero que se encuentra en un lugar distinto dentro de la estructura de directorios.

[11] Los demonios (*daemons*) no son más que procesos que se ejecutan en segundo plano. Estos demonios ejecutan diferentes funciones y proporcionan ciertos servicios, pero sin la interacción del usuario.

- **Correo electrónico:** Permiten recibir y enviar correos electrónicos.
- **Información de red:** Permiten centralizar la información de los ordenadores, usuarios y varios recursos de la red, facilitando la administración, de manera que estos no dependan de su situación en la red.
- **Servicios de nombres:** Permiten nombrar y traducir los diversos nombres por los que se conoce a un mismo recurso.
- **Servicios de acceso remoto:** Permiten interactuar desde el exterior con cualquier sistema brindando la posibilidad de ejecutar aplicaciones u obtener información remota de los mismos.
- **Servicios de generación de nombres:** Permiten en redes TCP/IP, una generación dinámica o estática de las direcciones IP que se disponen en función de las máquinas que las necesiten.
- **Servicios web:** Servicios que se brindan a través de un servidor web.
- **Servicios de acceso a Internet:** Permiten el acceso a Internet por medio de pasarelas (*gateways*) o por intermediarios (*proxys*) y no directamente.
- **Servicios de filtrado:** Permiten aplicar medidas de seguridad para filtrar información incorrecta o que afecten la seguridad de un determinado sistema.

1.3 Conexiones remotas

Una conexión remota es una operación realizada en un ordenador remoto a través de una red de ordenadores, como si se tratase de una conexión local. La conexión remota es importante para poder comunicarse desde un solo ordenador (servidor) hacia varios clientes, o así mismo tener conexión desde otro ordenador a un escritorio personal. Permiten ejecutar en un equipo remoto ciertos tipos de acciones desde un equipo local [10].

1.3.1 Conexiones remotas en GNU/Linux

Existen varias herramientas y protocolos que pueden ser utilizadas para realizar conexiones remotas en los sistemas GNU/Linux. A continuación se caracterizan algunas de ellas:

- **Telnet**: Es un protocolo estándar de Internet que permite conectar terminales y aplicaciones en Internet. Proporciona reglas básicas que permiten vincular a un cliente con un intérprete de comandos (del lado del servidor) [11]. Este protocolo posee varios problemas, dentro de ellos se encuentran:
    - Autentificación en texto plano.
    - Texto en claro de todos los comandos.
- **SSL Telnet:** Es Telnet con el añadido de Cifrado SSL (*Secure Sockets Layer*[12]), lo que lo hace más seguro. Usando certificados X.509 (también conocidos como certificados personales) se pueden administrar sistemas con facilidad. SSL Telnet es completamente libre y gratis para cualquier uso [11].
- **SSH** (*Secure Shell*): Es un protocolo de red, que permite establecer una comunicación a través de un canal seguro entre un cliente local y un servidor remoto. Utiliza una clave pública cifrada para autenticar el servidor remoto y permitir al servidor remoto autenticar el usuario. SSH provee confidencialidad e integridad en la transferencia de los datos utilizando criptografía y *Message Authentication Codes*[13]. De modo predeterminado, escucha peticiones a través del puerto 22 por TCP [11].
- **VNC** (*Virtual Network Computing*[14]): Es un programa de *software* libre basado en una estructura Cliente-Servidor que permite tomar el control del ordenador servidor remotamente a través de un ordenador cliente. También es llamado *software* de escritorio remoto. VNC no impone restricciones en el sistema operativo del ordenador servidor con respecto al del cliente. Por defecto, VNC no es un protocolo seguro [12].
- **NX:** Es una tecnología para manejar conexiones remotas a X Window. Para ello utiliza compresión de datos y mecanismos de *cache*, que le proporcionan un rendimiento netamente superior al de otras soluciones de este tipo como VNC. También emplea SSH para cifrar la conexión entre servidor y cliente. Además de permitir a los usuarios autenticarse en una máquina remota accediendo al escritorio, permite también suspender y recuperar sesiones. NX es un producto que dispone de licencia GPL, existiendo múltiples implementaciones, tanto comerciales como gratuitas, y tanto libres como propietarias, de servidores y clientes [13].

[12] Traducción de los autores: Capa de Conexión Segura.

[13] Traducción de los autores: Códigos de Autenticación de Mensaje.

[14] Traducción de los autores: Computación Virtual en Red.

- **Rlogin** (*Remote Login*): Es una aplicación TCP/IP que comienza una sesión de terminal remoto sobre el anfitrión especificado como *host*. El anfitrión remoto debe hacer funcionar un servicio de Rlogind (o demonio) para que el Rlogin conecte con el anfitrión. Utiliza un mecanismo estándar de autorización de los Rhosts [14].
- **Rexec:** Es una de las utilidades UNIX más antiguas la cual permite ejecutar comandos en un sistema remoto. Posee el serio fallo de no tener un modelo de seguridad real. La seguridad se consigue mediante el uso de ficheros “Rhosts”, que especifican qué ordenadores pueden ejecutar comandos, lo cual está sujeto a *Spoofing* y *Exploits* [14].
- **Fsh:** "Ejecución rápida de comandos remotos", evita el costo de estar creando continuamente sesiones cifradas, habilitando un túnel cifrado utilizando SSH o LSH, sobre el cual ejecuta todos los comandos remotos [14].
- **OpenSSH**: Es una implementación gratuita del protocolo SSH, con licencia BSD [15].
- **Dropbear:** Es una implementación ligera del protocolo SSH versión 2. Es un *software* de código abierto. Dentro de sus principales características se encuentran [16]:
    - Posee bajo consumo de memoria.
    - Implementa el reenvío por X11 y el reenvío de la autenticación de agente para los clientes de OpenSSH.
    - Es compatible con el reenvío de TCP.
- **Putty:** Es un cliente gráfico que permite realizar conexiones remotas a través de los protocolos SSH y Telnet. Dentro de sus principales características se encuentran [17]:
    - Permite el almacenamiento de *hosts* y preferencias para uso posterior.
    - Posee un control sobre la clave de cifrado SSH y la versión de protocolo.
    - Permite el redireccionamiento de puertos con SSH, incluyendo manejo empotrado de reenvío X11.
    - Posee soporte para los algoritmos de cifrado 3DES, AES, RC4, Blowfish, DES.
    - Posee soporte para la autenticación con claves públicas.

- **Vinagre:** Es una aplicación de escritorio que permite realizar conexiones remotas de forma gráfica, integrado al entorno de escritorio Gnome. Soporta los protocolos VNC y SSH [18].
- **Vino:** Es un servidor para el protocolo VNC integrado al entorno de escritorio Gnome [19].
- **XDMCP** (X *Display Manager Control Protocol*): Es un protocolo utilizado en redes para comunicar un ordenador servidor que ejecuta un sistema operativo con un gestor de ventanas basado en X Window [20].
- **RealVNC:** Consta de un servidor y una aplicación cliente para el protocolo VNC [21].
- **LSH:** Implementación gratuita del protocolo SSH. LSH tiene licencia GNU. Uno de los objetivos de LSH es que sea razonable, fácil de extender, sin jugar con las funciones de seguridad básicas [22].

1.4 Tecnologías a utilizar

Con el objetivo de disminuir los costos, el tiempo utilizado y obtener mejores resultados, desde el inicio del desarrollo de un *software* es necesario seleccionar las tecnologías adecuadas a usar.

Esta selección debe realizarse teniendo en cuenta que Smart Keeper es una aplicación disponible únicamente para los sistemas operativos GNU/Linux, así como las características, funcionalidades y ventajas que reportaría el uso de cada una de las herramientas seleccionadas.

1.4.1 Selección de tecnologías para la conexión remota

Después de haber analizado cada uno de los protocolos y herramientas caracterizadas se selecciona SSH como el protocolo para las conexiones remotas. Esta selección se realiza teniendo en cuenta la gran ventaja que SSH posee sobre las demás protocolos que realizan conexiones remotas: con SSH la información va completamente cifrada. Esta ventaja le provee gran seguridad a cada una de las conexiones. A continuación se abordan las principales características y métodos de autenticación que posee dicho protocolo [23].

SSH permite:

- Iniciar sesiones (*login*) en servidores remotos.

- Ejecutar comandos remotamente.
- Copiar archivos entre distintos ordenadores.
- Ejecutar aplicaciones X11 remotamente.
- Realizar túneles IP cifrados.

Métodos de autenticación de usuarios del protocolo SSH

- **Autenticación con contraseña:** Es el método más simple de autenticación. El cliente solicita al usuario el ingreso de una contraseña y la misma es enviada al servidor, el cual validará la misma utilizando los mecanismos configurados en el sistema. Generalmente, utilizará la autenticación del sistema Unix para validar la contraseña contra el contenido del archivo ***/etc/shadow.***

  Las desventajas de este método de autenticación son:

  - El usuario debe escribir su contraseña cada vez que se conecta al servidor.
  - La contraseña del usuario es enviada hacia el servidor.

- **Autenticación con clave pública:** Para poder realizarse, el usuario debe tener un par de claves pública/privada, y la clave pública debe estar almacenada en el servidor. Luego de establecida la conexión, el servidor genera un número aleatorio llamado desafío, que es cifrado con la clave pública del usuario usando RSA o DSA. El texto cifrado es enviado al cliente, que debe descifrarlo con la clave privada correspondiente y devolverlo al servidor, demostrando de esta manera que el usuario es quien dice ser.

  Las ventajas de este método de autenticación respecto al anterior son:

  - El usuario no debe escribir (ni recordar) ninguna contraseña.
  - La clave privada del usuario es enviada al servidor.

Para el desarrollo de la solución se utilizará el método básico de autenticación, a través de contraseña, esto facilitará el trabajo del sistema con múltiples administradores y servidores donde no se tendrán que generar llaves privadas y públicas.

1.4.2 Implementación para el protocolo SSH

Como implementación para el protocolo SSH se selecciona OpenSSH teniendo en cuenta su seguridad y disponibilidad en los repositorios de *software*. A continuación se exponen sus principales características:

Características [24]:

- **Proyecto de código abierto y licencia libre:** El código fuente de OpenSSH está disponible para todo aquel que desee obtenerlo en Internet. Se puede usar para cualquier propósito, y esto incluye su uso comercial.
- **Cifrado fuerte:** OpenSSH soporta 3DES, Blowfish, AES y Arcfour como algoritmos de cifrado.
- **Autenticación fuerte:** Una fuerte autenticación protege contra varios problemas de seguridad, como por ejemplo suplantación de IP, rutas falsas y fisgoneo de DNS.
- **Compresión de datos:** La compresión de datos antes del cifrado mejora los resultados en los enlaces con redes lentas.
- **Reenvío por agente:** Un agente de autenticación que se encuentre en la estación de trabajo o el portátil de un usuario, se puede usar para contener las claves de autenticación de RSA o DSA. OpenSSH envía la conexión automáticamente al agente de autenticación por medio de cualquier conexión y de este modo no existe la necesidad de guardar las claves de autenticación de RSA o DSA en ninguna máquina de la red exceptuando la máquina del usuario. Los protocolos de autenticación nunca revelan las claves; sólo se pueden usar para verificar que el agente del usuario tenga cierta clave.
- **Reenvío por puertos:** Permite enviar conexiones de TCP/IP a una máquina remota por un canal cifrado.

1.4.3 Lenguajes de programación

Un lenguaje de programación actúa como un traductor entre el usuario y el equipo. En lugar de aprender el lenguaje nativo del equipo conocido como lenguaje máquina, se puede utilizar un lenguaje de programación para dar instrucciones al equipo de un modo que sea más fácil de aprender y entender.

Teniendo en cuenta que el sistema Smart Keeper está desarrollado utilizando PHP versión 5 se selecciona este lenguaje para el desarrollo del módulo, para así poder integrarlo correctamente al sistema. A continuación se describen sus principales características:

Características [25]:

- Es un lenguaje multiplataforma.
- Permite las técnicas de Programación Orientada a Objetos.
- Posee una biblioteca nativa de funciones sumamente amplia.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Completamente orientado a la web.
- Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros.
- Posee una amplia documentación.

A continuación se exponen los lenguajes auxiliares utilizados para el desarrollo del módulo:

HTML (*Hypertext Markup Language*[15])**:** Es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto. Gracias a Internet, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos. Este lenguaje se utilizará en la creación de la interfaz web [26].

Bash: Acrónimo de *Bourne-Again Shell*, es un lenguaje de programación interpretado, el cual permite la ejecución de comandos. Es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux [27]. Este lenguaje será utilizado para gestionar toda la ejecución de comandos durante el proceso de configuración de cada uno de los servicios.

[15] Traducción de los autores: Lenguaje de formato de documentos de hipertexto.

1.4.4 Bibliotecas para la conexión SSH a través de PHP

Para poder realizar las conexiones remotas usando el protocolo SSH y el lenguaje PHP es necesario la utilización de una biblioteca. Para este lenguaje existen dos bibliotecas fundamentales que permiten realizar estas acciones, phpseclib y libssh2-php. A continuación se describen ambas:

- **libssh2-php**: Es una biblioteca escrita en lenguaje de programación C mediante la cual permite realizar conexiones SSH a través de PHP. Está disponible para todas las versiones de este lenguaje de programación. La misma permite:
  - Autenticación utilizando una llave de host público, una clave pública o con una contraseña simple.
  - Ejecutar un comando en un servidor remoto.
  - Recuperar una secuencia de datos ampliada y las huellas dactilares del servidor remoto.
  - Añadir una clave pública autorizada.
  - Enviar y obtener fichero mediante SCP.
  - Crear enlaces simbólicos y directorios.
  - Renombrar y eliminar ficheros remotos.
  - Abrir túneles a través de un servidor remoto y solicitar un *shell* interactivo.

Está disponible en todos los repositorios de *software* de Debian GNU/Linux, su documentación viene incluida en los manuales oficiales de PHP [28].

- **phpseclib**: Es una biblioteca completamente escrita en PHP, reuniendo gran cantidad de características las cuales se encuentran implementadas en PHP a través de extensiones. Posee implementación para los algoritmos criptográficos AES, DES, RSA, Rijndael, RC4 y TripleDES. Contiene además funciones para realizar conexiones remotas a través de SSH, permitiendo la autenticación a través de claves públicas, ejecutar comandos en el servidor remoto, así como también obtener y enviar ficheros. Solo está disponible para PHP versión 4 [29].

Teniendo en cuenta sus funcionalidades, documentación y disponibilidad se selecciona la biblioteca **libssh2-php** para realizar las conexiones remotas a través del lenguaje de programación PHP.

1.4.5 Gestión de acceso administrativo en servidores

Por cuestiones de seguridad en ocasiones la conexión remota no debe realizarse a través de un usuario con privilegios administrativos del sistema, por lo que es necesario utilizar una aplicación que le brinde estos permisos luego de realizada la conexión.

Sudo: Aplicación que le brinda a un usuario acceso Setuid a un programa, se le puede especificar desde qué ordenadores se les permite (o no) hacer *login* y tener acceso. Se puede especificar bajo qué usuario se ejecutará un comando, lo cual provee un grado de control relativamente preciso. Sudo está disponible para la mayoría de las distribuciones, como paquete interno, o paquete contribuido [30]. Este comando se utilizará para gestionar el acceso como administrador en los servidores remotos en caso de que este lo requiera.

1.4.6 Herramientas de programación

Los entornos de desarrollo integrado (IDE, por sus siglas en inglés) poseen grandes ventajas que contribuyen al ahorro de tiempo. Estas radican fundamentalmente en la automatización de algunas tareas asociadas al desarrollo de aplicaciones, así como el soporte para algunas funcionalidades tales como el autocompletamiento de código, depuración, formateo y el trabajo con el subversión. La selección de los IDEs con soporte para PHP se centró en tecnologías libres teniendo en cuenta las funcionalidades, facilidades y experiencia en el trabajo con los mismos. Dentro de los principales candidatos se obtuvieron Eclipse, Zend Studio, Bluefish, Komodo, y NetBeans, seleccionándose este último por sus funcionalidades y características:

NetBeans IDE es un entorno de desarrollo integrado, el cual permite que los desarrolladores puedan escribir, compilar, depurar y ejecutar programas [31]. Algunas de sus principales características son:

- Producto de código abierto.
- Gran base de usuarios y una comunidad de desarrolladores en constante crecimiento.
- Conjuntos de herramientas independientes de la plataforma y modulares.
- Facilidad de uso.
- Cumplimiento de regulaciones.

- Flexibilidad entre plataformas.

1.4.7 *Framework* de desarrollo

Un *framework* permite reutilizar un diseño para un dominio específico de *software* rigiendo la arquitectura del mismo. Además, define la estructura global de un sistema teniendo en cuenta las clases y objetos, sus responsabilidades claves y como colaboran entre sí, de modo que los diseñadores e implementadores puedan concentrarse en los elementos específicos del mismo.

Para la selección del *framework* de desarrollo se tuvo en cuenta que Smart Keeper, está desarrollado usando Symfony versión 1.2, por lo que se continuará utilizando este mismo *framework,* con el objetivo fundamental de integrar el resultado final al sistema.

Symfony v1.2: Es un *framework* diseñado para optimizar el desarrollo de las aplicaciones web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Symfony está desarrollado completamente con PHP versión 5 [32].

Características de Symfony v1.2:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además es lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

Módulos en Symfony v1.2

En ocasiones, es necesario reutilizar una porción de código desarrollada para alguna aplicación Symfony. Si se puede encapsular ese código en una clase, tan sólo es necesario guardar la clase en algún directorio ***lib/*** para que otras aplicaciones puedan encontrarla. Sin embargo, si el código se encuentra desperdigado en varios archivos, como por ejemplo un tema para el generador de administraciones o una serie de archivos JavaScript y *helpers* que permiten utilizar fácilmente un efecto visual complejo, es muy complicado copiar todo este código en una clase.

Los módulos permiten agrupar todo el código diseminado por diferentes archivos y reutilizar este código en otros proyectos. Los módulos pueden encapsular clases, filtros, *mixins*, *helpers*, archivos de configuración, tareas, módulos, esquemas y extensiones para el modelo, *fixtures* o archivos estáticos. Los módulos son fáciles de instalar, de actualizar y de desinstalar. Se pueden distribuir en forma de archivo comprimido .tgz, un paquete PEAR o directamente desde el repositorio de código. La ventaja de los paquetes **PEAR** es que pueden controlar las dependencias, lo que simplifica su actualización. La forma en la que Symfony carga los módulos permite que los proyectos puedan utilizarlos como si fueran parte del propio *framework*.

Básicamente, un módulo es una extensión encapsulada para un proyecto Symfony. Estos permiten no solamente reutilizar código propio, sino que aprovechan los desarrollos realizados por otros programadores y añaden al núcleo de Symfony extensiones realizadas por otros desarrolladores.

Sobre la base de esto se concluye que en aras de desarrollar una aplicación que se integre a Symfony versión 1.2 de manera fácil y que además sea flexible a futuras modificaciones, se desarrollará un módulo con las funcionalidades requeridas para la solución.

1.5 Metodología de desarrollo

Las metodologías de desarrollo de *software* abarcan todo el ciclo de vida del *software*, y se definen como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que guían el proceso de creación de aplicaciones. Los procedimientos detallan consejos para elaborar una actividad; las técnicas serían la forma de ejecutar un procedimiento para obtener un resultado determinado; las herramientas de *software* son las que hacen posible automatizar el proceso de desarrollo del *software* y la documentación es la que identifica el *software* que se está desarrollando [33].

Extreme programming[16] **(XP)**

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP es adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [34].

Características de la metodología XP

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación en parejas.
- Frecuente interacción del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad sin modificar su comportamiento.

Scrum

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en: El desarrollo de *software* se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas se destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración [35].

[16] Traducción de los autores: Programación extrema.

En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad y la productividad son fundamentales.

SXP (Scrum + XP)

Es una metodología compuesta por las metodologías SCRUM y XP que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de *software* para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo [36].

Rational Unified Process[17] (RUP)

Es un proceso de desarrollo de *software* que define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo, además es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto [37]. Dentro de las principales características y las más significativas de este proceso se encuentran:

- **Dirigido por Casos de Uso:** Un *software* se crea para servir a sus usuarios, por lo tanto, para construir un sistema exitoso se debe conocer qué es lo que quieren y necesitan los usuarios prospectos introduciéndose así, los casos de uso (CU) que es donde se definen los requisitos funcionales (RF), que el sistema debe cumplir [37].
- **Centrado en la arquitectura:** La arquitectura en un sistema de *software* es descrita desde los distintos puntos de vista del mismo, involucrando los aspectos estáticos y dinámicos más significativos [37].
- **Iterativo e incremental:** El ser iterativo e incremental le permite al equipo de desarrollo en un tiempo estimado obtener una pequeña parte del producto de acuerdo con una fase que pase por todas las disciplinas, esto se conoce por iteración, permite además según cada iteración un crecimiento del producto, conocido como incremento [37].

[17] Traducción de los autores: Proceso Unificado de Rational.

Principios de desarrollo de RUP

A continuación se muestran los 5 principios claves [37] en los que está basado RUP:

- **Adaptar el proceso:** El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.
- **Balancear prioridades:** Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.
- **Demostrar valor iterativamente:** Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.
- **Elevar el nivel de abstracción:** Este principio dominante motiva el uso de conceptos reutilizables. Esto previene a los ingenieros de *software* ir directamente de los requisitos a la codificación de *software* a la medida del cliente. Un nivel alto de abstracción también permite discusiones sobre diversos niveles arquitectónicos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.
- **Enfocarse en la calidad**: El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Del estudio anterior se selecciona RUP como proceso rector por ser el que más se adapta a las características de este proyecto de *software*. Esta metodología hace énfasis en realizar una buena captura de los requisitos y genera una amplia documentación, cuestiones que no son abordadas de esta manera por las demás metodologías expuestas. Posee evaluación en cada iteración que permite cambios de objetivos. Sigue los pasos intuitivos necesarios a la hora de desarrollar el *software* además del seguimiento detallado en cada una de las iteraciones. Otra de las razones de peso en la elección es sin lugar a dudas el conocimiento y familiarización del equipo de desarrollo con esta metodología.

1.5.1 Lenguaje de modelado y herramienta CASE

Como lenguaje de modelado se selecciona UML; este permite visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo. Además, permite la modelación de sistemas con tecnologías orientada a objetos.

Para apoyar a la metodología RUP seleccionada se escoge como herramienta CASE el Visual Paradigm. Esta selección se realiza considerando que esta herramienta contiene una buena semántica y una organización dirigida a diagramas lo cual permite una mejor organización de la documentación. También permite generar código y documentar los diagramas. Además, presenta una fuerte integración con el gestor de base de datos PostgreSQL.

1.6 Conclusiones

En este capítulo se abordaron todos los elementos teóricos que sustentan la solución del problema llegando a las siguientes conclusiones:

- Debido a la no existencia de una aplicación que de solución a la situación problemática es necesario la creación de un módulo que cumpla con todas las necesidades de Smart Keeper.
- Dada la seguridad que provee el protocolo SSH, se seleccionó para realizar las conexiones a los servidores remotos.
- Debido a que el sistema Smart Keeper está completamente desarrollado en PHP usando el *framework* Symfony y las ventajas que estos poseen para el desarrollo de aplicaciones web, fueron seleccionados para el desarrollo de la solución.

Capítulo 2

Características del sistema

Introducción

El desarrollo de un *software* parte de comprender la problemática a la que se quiere dar solución. En RUP la fase de elaboración permite lograr lo anterior a través del levantamiento de requisitos y la realización del Modelo de Negocio o Dominio [38]. Para la propuesta solución se partirá de la descripción del problema y se planteará como estará estructurada esta, dando lugar a los requisitos funcionales y no funcionales. Por último, se hará una descripción de los casos de usos y se presentarán los diagramas asociados a estos.

2.1 Propuesta de solución

En el capítulo anterior se presentó la necesidad de crear un módulo que se integre a la interfaz de administración web de Smart Keeper. El objetivo del mismo es permitir la gestión de cada uno de los servicios que componen el sistema de forma eficiente y sencilla, pero puede surgir la siguiente interrogante: ¿cuáles son los servicios que integran Smart Keeper? Para dar respuesta a la misma es preciso profundizar en el funcionamiento de dicho sistema.

2.1.1 Servicios a configurar

Smart Keeper está compuesto por una variedad de servicios que hacen necesaria la existencia de una herramienta integrada a la interfaz de administración web de Smart Keeper que permita gestionarlos de forma fácil e intuitiva. Para lograr esto a continuación se hará un estudio de los principales servicios que lo componen:

Apache v2

Es un servidor web de código abierto, multiplataforma y muy popular actualmente. Es usado para muchas tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Algunas de sus principales características son [39]:

- Es un servidor web flexible, rápido y eficiente.
- Multiplataforma.
- Tecnología de código fuente abierto.
- Altamente configurable y de diseño modular.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.

Estructura de directorios

Apache ubica sus principales ficheros de configuración en [40]:

/etc/apache2/

apache2.conf	Fichero de configuración principal de Apache.
http.conf	Fichero de tipo ASCII que contiene directivas de configuración.

Por defecto almacena los *logs* del servicio en:

/var/log/apache2/error.log

PostgreSQL v8.4

Es un robusto programa de gestión de base de datos relacional orientado a objetos y libre, publicado bajo licencia BSD[18]. Ofrece control de concurrencia multiversión, soporta casi toda la sintaxis SQL, cuenta con un amplio conjunto de enlaces con lenguajes de programación, son algunas de sus principales características entre otras [41]:

- Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los

[18] Licencia BSD: Licencia de *software* otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de Software Libre.

gestores objeto-relacionales.

- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Alta concurrencia, PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Soporte para vistas, claves foráneas, integridad referencial, disparadores, procedimientos almacenados y subconsultas.

Estructura de directorios

PostgreSQL ubica sus principales ficheros de configuración en [42]:

/etc/postgresql/8.4/main/

pg_hba.conf	Se utiliza para definir los diferentes tipos de accesos que un usuario tiene en el clúster.
pg_ident.conf	Se utiliza para definir la información necesaria en el caso que utilicemos un acceso del tipo *ident* en *pg_hba.conf*.
postgresql.conf	Contiene todos los parámetros de configuración que afectan al funcionamiento y al comportamiento de PostgreSQL

Por defecto almacena los *logs* del servicio en:

/var/log/postgresql/postgresql-8.4-main.log

Clamav

Uno de los pocos antivirus de código abierto, bajo licencia GPL. ClamAV incluye un escáner que detecta más de 720 000 virus, gusanos y troyanos, y cuya base de datos se actualiza automáticamente por Internet. Posee la capacidad para examinar contenido de ficheros ZIP, RAR, Tar, Gzip, Bzip2, MS OLE2, MS Cabinet, MS CHM y MS SZDD [43].

Estructura de directorios

Clamav ubica sus principales ficheros de configuración en [44]:

/etc/clamav/

clamav.conf	Contiene parámetros globales de ClamAV, como los serían generación de registros y límites de archivos a inspeccionar.
freshclam.conf	Incluye la configuración para consultar la base de datos ClamAV y así actualizar la información local referente a virus más recientes

Por defecto almacena los *logs* del servicio en:

/etc/clamav/

clamav.log	*Logs* del servicio.
freshclam.log	*Logs* de la base de datos del servicio.

Protocolo ICAP. GreasySpoon

ICAP es un protocolo de red abierto y público originado en 1999 para la redirección de contenidos con fines de filtrado y conversión. Permite el uso de antivirus, filtrado de contenidos, traducción dinámica de páginas, inserción automática de anuncios, compresión de HTML, entre otros. GreasySpoon es una implementación de este protocolo inspirado por la extensión de Firefox Greasemonkey[19], permite manipular el tráfico HTTP mediante la creación de secuencias de comandos simples en varios idiomas posibles. Es una solución simple y eficaz para interceptar, filtrar y transformar el tráfico de Internet sobre la marcha [45].

Estructura de directorios

GreasySpoon ubica sus principales ficheros de configuración en:

/usr/local/fcweb/icap/conf/

icapserver.conf	Fichero de configuración principal de GreasySpoon.

Por defecto almacena los *logs* del servicio en:

/usr/local/fcweb/icap/log/error.log

Squid v3

[19] Disponible en: <https://addons.mozilla.org/es-ES/firefox/addon/748>.

El *proxy cache* Squid es una excelente solución para optimizar el uso del enlace a Internet y acelerar el tráfico web ya que almacena los contenidos más frecuentemente accedidos. Brinda además mecanismos muy flexibles para administrar el acceso por usuarios, equipos, URLs, tipos de contenidos y demás. Soporta distintos filtros de contenido, algunos de los cuales permiten la utilización de "listas negras" que contienen listados de sitios clasificados por categoría [46].

Estructura de directorios

Squid ubica sus principales ficheros de configuración en [47]:

/etc/squid3/

- *squid.conf* Es el fichero principal de configuración de Squid.

Por defecto almacena los *logs* del servicio en:

/var/log/squid3/cache.log

2.1.2 Estructura de instalación del sistema Smart Keeper

Con el fin de ofrecer la posibilidad de concentrar *hardware* especializado según las necesidades de algún servicio en particular el sistema Smart Keeper, puede ser instalado de forma distribuida en diferentes nodos de la red [48], como se muestra en la Figura 1 del Anexo A. Esto logra un mejor rendimiento del sistema y hace frente al elevado consumo de recursos computacionales cuando todos los servicios coinciden en un mismo nodo, pero hace más complicado para el administrador gestionar todos los servicios, pudiendo este llegar a tener hasta 5 servidores bajo su responsabilidad y en el peor de los casos en locales distintos.

2.1.3 Características de la solución

En el capítulo anterior se seleccionó como *framework* de desarrollo a Symfony con el objetivo fundamental de integrar el resultado final al sistema. También se estudió entre las ventajas de Symfony que permite la creación de módulos para encapsular clases, filtros, *mixins*, *helpers*, archivos de configuración, tareas, módulos, esquemas y extensiones para el modelo, *fixtures* o archivos estáticos. Esto es muy útil para adicionar funcionalidades a los sistemas sin tener que reescribir todo el código. Para el caso de la solución

propuesta en este trabajo se aprovechará esta ventaja para desarrollar un módulo que pueda ser integrado al sistema Smart Keeper.

El módulo estará enfocado en permitirle al administrador gestionar los servicios que componen a Smart Keeper desde su propia interfaz de administración. Este estará compuesto por otros 7 submódulos principales, uno principal para mostrar y cambiar el estado de los servicios activos y cada uno de los 6 restantes estará dedicado a un servicio específico del sistema.

La siguiente ilustración muestra una idea general de la solución propuesta:

Figura 1: Módulo propuesto.

El módulo se adicionará a la interfaz añadiéndose en el menú principal bajo una pestaña con el nombre “Servicios”. Al activarse el módulo se listarán los servicios activos y el estado en que se encuentran cada uno de ellos. Además, se presentará un submenú que permitirá acceder a cada uno de los servicios a gestionar. Para cada uno de estos servicios se permitirá realizar las tareas de administración concernientes.

La conexión entre el cliente y el servidor web Apache que soporta la interfaz de administración se hará

utilizando el protocolo seguro HTTPS provisto por el propio servidor. Como se definió en el capítulo 1, para el caso de las conexiones a cada uno de los servicios que integran a Smart Keeper se utilizará el protocolo SSH a través de la biblioteca libssh2-php del lenguaje de programación PHP también definido anteriormente.

2.2 Modelo de dominio

Existen varias alternativas para llevar a cabo el modelado de un sistema, RUP, como metodología de desarrollo propone la realización de un modelo de negocio para el caso en el que los procesos dentro del entorno estén claramente definidos. Además, propone la realización de un modelo de dominio para los escenarios en los que no puedan identificarse tales procesos del negocio [49].

Después de haber realizado un estudio de los procesos que se van a efectuar, se determina que el negocio estudiado tiene muy bajo nivel de estructuración, donde los flujos de información se encuentran difusos, además es difícil establecer las reglas de funcionamiento, por lo que se propone realizar un modelo de dominio.

A continuación se describen cada una de las clases que componen al Modelo de Dominio:

Administrador: Persona encargada de realizar todas las acciones en el Módulo para la gestión remota de los servicios que componen a Smart Keeper.

Interfaz Web: Interfaz con la que interactúa el administrador y mediante la cual se realizan todas las acciones.

Servicios: Representa los servicios que se gestionan.

Ficheros de *Logs*: Son los ficheros de *logs* que se le asignan a cada servicio.

Ficheros de Configuración: Son los ficheros de configuración asociados a cada servicio.

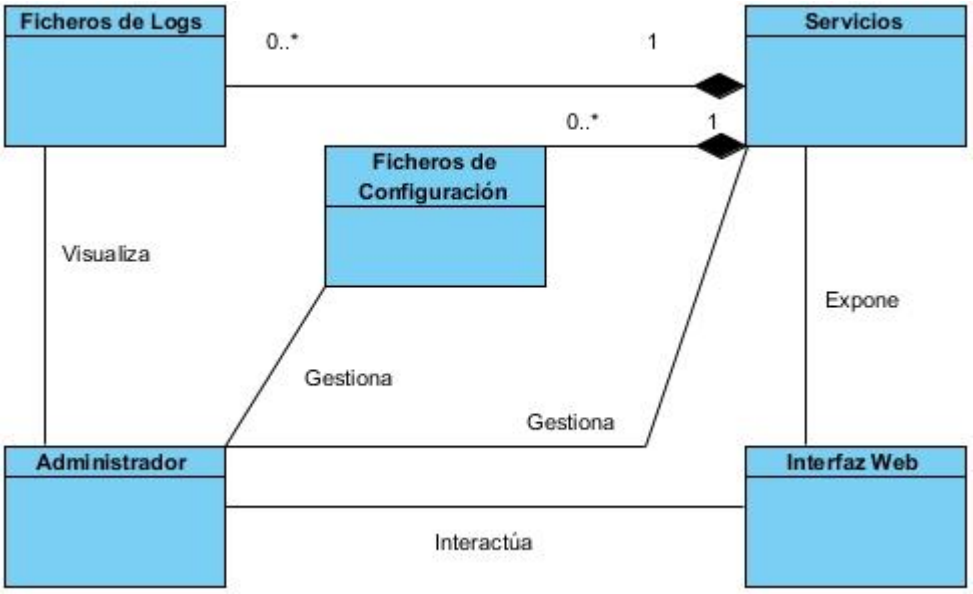


Figura 2: Diagrama de Clases del Modelo del dominio.

2.3 Modelado del sistema

2.3.1 Especificación de los requisitos del *software*

El flujo de trabajo de requerimientos es uno de los más importantes, porque en él se establece qué es lo que tiene que hacer exactamente el sistema que se construya. En esta línea los requisitos son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen. Se dividen en dos grupos: los requisitos funcionales y los requisitos no funcionales [50].

El proceso de reunión de requisitos se intensifica y se centra específicamente en el *software*. Para comprender la naturaleza del (los) programa(s) a construirse, el ingeniero (analista) del *software* debe comprender el dominio de la información del *software*, así como la función requerida, comportamiento, rendimiento e interconexión.

2.3.2 Requisitos funcionales

Una vez descrito el modelo de dominio, para poder identificar que debe hacer el sistema y entender su funcionamiento, es fundamental conocer los requisitos funcionales que el sistema debe cumplir.

A continuación se relacionan los requisitos funcionales identificados:

RF1: Adicionar servicios.

- El sistema debe permitir adicionar servicios como: PostgreSQL, Apache, Clamav, Redirector, Squid y GreasySpoon.

RF2: Eliminar servicios.

- El sistema debe permitir eliminar los servicios que han sido adicionados.

RF3: Editar servicios.

- El sistema debe permitir editar los servicios que han sido adicionados.

RF4: Adicionar ficheros de configuración.

- El sistema debe permitir adicionar ficheros de configuración a los servicios.

RF5: Eliminar ficheros de configuración.

- El sistema debe permitir eliminar los ficheros de configuración adicionados a los servicios.

RF6: Editar ficheros de configuración.

- El sistema debe permitir editar los ficheros de configuración adicionados a los servicios.

RF7: Configurar servicios.

- El sistema debe permitir mostrar los ficheros de configuración para ser editados.

RF8: Iniciar los servicios.

- El sistema debe permitir iniciar los servicios.

RF9: Detener los servicios.

- El sistema debe permitir detener los servicios.

RF10: Restaurar los servicios.

- El sistema debe permitir restaurar los servicios.

RF11: Reiniciar los servicios.

- El sistema debe permitir reiniciar los servicios.

RF12: Mostrar estado de los servicios.

- El sistema debe permitir mostrar el estado de los servicios.

RF13: Mostrar los *logs* de los servicios.

- El sistema debe permitir mostrar los *logs* de los servicios.

RF14: Mostrar listado de servicios activos.

- El sistema debe permitir mostrar el listado de los servicios activos.

2.3.3 Requisitos no funcionales

Una vez analizadas las funcionalidades que el sistema debe cumplir se hace necesario analizar las propiedades que el producto de *software* debe tener [51]. A esto se le conoce como requisitos no funcionales.

RNF 1. Apariencia o interfaz externa:

- La interfaz debe ser sencilla, intuitiva y amigable para sus usuarios, acorde a la interfaz actual de la aplicación web Smart Keeper.
- El sistema permitirá visualizar el contenido de la información de forma organizada y legible.

RNF 2. Usabilidad:

- Deberá poseer una interfaz y navegación asequible y funcional tanto para usuarios expertos como para los que no tienen conocimientos profundos del sistema.

RNF 3. Eficiencia:

- El sistema deberá tener un nivel de respuesta aceptable, tanto para los accesos a la base de datos, como para el proceso de administración de tareas.

RNF 4. Seguridad:

- El acceso al módulo se realizará mediante una conexión segura por HTTPS y la autenticación de los usuarios mediante los mecanismos de autenticación de Smart Keeper. La seguridad se establecerá por roles que se le asignarán a los usuarios que interactúen con el sistema.

RNF 4. Confiabilidad:

- El sistema debe tener soporte para recuperación ante fallos y errores.

RNF 5. *Software*:

- Sistema operativo Debian GNU/Linux v6.

RNF 6. *Hardware*:

- El servidor web debe contar con al menos 1 Gb de memoria RAM.
- El servidor web debe poseer un procesador Intel o similar, a 3.0 GHz o superior.

RNF 7. Administración:

- La herramienta debe estar bien organizada para facilitar su administración. En caso de alguna modificación en la programación debe permitir hacerse en el paquete correspondiente sin afectar a los demás.

2.3.4 Definición de los actores del sistema

Los actores del sistema intercambian información con él, aunque no forman parte de este. Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado [51].

A continuación se muestra el actor y la justificación que tiene en el sistema:

Actor	Administrador
Justificación	Persona que interactúa con el Módulo para la gestión remota de los servicios. Es el responsable de administrar los servicios que componen a Smart Keeper y gestionar su estado.

Tabla 1: Definición del actor del sistema.

2.3.5 Definición de los casos de uso del sistema

Los casos de uso son fragmentos de funcionalidades que agrupan los requisitos que debe cumplir el sistema. El modelo de casos de uso describe las funciones que realiza el sistema y la interacción de estas con el actor [51]. En el sistema que se modela se definen los siguientes casos de uso:

CU1	Gestionar Servicios
Actor	Administrador
Descripción	Es el caso de uso que da la posibilidad de poder insertar, eliminar y modificar un servicio en el sistema.

Tabla 2: Definición del caso de uso - Gestionar Servicios.

CU2	Gestionar ficheros de configuración
Actor	Administrador
Descripción	Es el caso de uso que da la posibilidad de poder insertar, eliminar y modificar un fichero de configuración a un servicio en el sistema.

Tabla 3: Definición del caso de uso - Gestionar ficheros de configuración.

CU3	Cambiar estado de los servicios
Actor	Administrador
Descripción	Es el caso de uso que da la posibilidad de poder iniciar, detener, restaurar y reiniciar un servicio en el sistema.

Tabla 4: Definición del caso de uso - Cambiar estado de los servicios.

CU4	Configurar servicios
Actor	Administrador
Descripción	Es el caso de uso que da la posibilidad de poder configurar un servicio en el sistema a través de sus ficheros de configuración.

Tabla 5: Definición del caso de uso - Configurar servicios.

CU5	Mostrar listado de servicios activos
Actor	Administrador
Descripción	Es el caso de uso que muestra un listado de todos los servicios activos en el sistema.

Tabla 6: Definición del caso de uso - Mostrar listado de servicios activos.

CU6	Mostrar *logs* de los servicios
Actor	Administrador
Descripción	Es el caso de uso que muestra los *logs* de los servicios en el sistema.

Tabla 7: Definición del caso de uso - Mostrar *logs* de los servicios.

2.3.6 Diagrama de casos de uso del sistema

A continuación se muestra el diagrama de casos de uso del sistema.

Figura 3: Diagrama de casos de usos del sistema.

2.3.7 Descripción expandida de los casos de uso del sistema

Para entender la funcionalidad asociada a los casos de uso no es suficiente con la representación gráfica del Diagrama de casos de uso. La descripción extendida brinda los detalles de la secuencia de las acciones, las precondiciones como estado inicial, los posibles estados finales como poscondiciones, además de cuando comienza y termina el caso de uso [51].

Describe explícitamente que debe hacer el sistema, separando las responsabilidades del sistema y la de los actores.

A continuación se muestra la descripción expandida de uno de los casos de uso del sistema, los demás se encuentran en el Anexo B.

Caso de Uso	Gestionar Servicios
Actores	Administrador
Resumen	El caso de uso se inicia cuando el administrador necesita adicionar un nuevo servicio, para ello necesita llenar los campos correspondientes al servicio.
Precondiciones	

Referencias	RF1,RF2,RF3
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "General"	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona en el menú principal "Servicios" y se dirige al submenú del servicio que desee.	2. El sistema muestra el listado de todos los servicios del tipo seleccionado.
Sección "Adicionar Servicio"	
1. El administrador selecciona la opción "Adicionar Servicio".	2. El sistema muestra una interfaz con los datos requeridos para crear un nuevo servicio.
3. El administrador introduce los datos y presiona el botón "Guardar".	4. El sistema valida los datos e inserta el nuevo servicio. El sistema muestra el mensaje "Elemento creado correctamente."
Flujos Alternos	
4.1 En caso de ocurrir algún error en los datos entrados el sistema muestra el mensaje "Elemento no guardado, se ha producido algún error.", inmediatamente se vuelve al paso 2.	
Sección "Eliminar Servicio"	
1. El administrador presiona el botón "Borrar", en la fila del servicio que desea eliminar.	2. El sistema muestra un diálogo para verificar si realmente se desea eliminar el servicio con las opciones Cancelar y Borrar.
3. El administrador presiona el botón "Borrar"	4. El sistema elimina el servicio y muestra el mensaje "Elemento borrado correctamente."
Flujos Alternos	
3.1 En caso de presionar el botón "Cancelar" no se realiza nada.	
Sección "Editar Servicio"	
1. El administrador selecciona la opción "Editar", en la fila del servicio que desee editar.	2. El sistema muestra una interfaz con todos los datos del servicio con la posibilidad de ser editados.

3. El administrador edita los datos que desee. 4. Presiona el botón “Guardar”.	5. El sistema valida los datos y guarda el servicio editado e inmediatamente muestra el mensaje “Elemento actualizado correctamente.”.
Flujos Alternos	
5.1 En caso de ocurrir algún error en los datos entrados el sistema muestra el mensaje “Elemento no guardado, se ha producido algún error.”, inmediatamente se vuelve al paso 2.	

Tabla 8: Descripción del caso de uso - Gestionar Servicios.

2.4 Conclusiones

A través de la obtención de los requisitos funcionales se pudo identificar lo que debe hacer el sistema y entender su funcionamiento. Además, los requisitos no funcionales permitieron definir los atributos que debe exhibir el sistema. Mediante la descripción detallada de los casos de uso se pudieron definir los detalles internos sobre qué debe hacer el sistema como respuesta a las acciones del actor.

Al concluir el presente capítulo se han creado las condiciones para efectuar el diseño y la implementación del módulo para la gestión remota de los servicios que componen a Smart Keeper.

Capítulo 3

Diseño del sistema

Introducción

El Modelo de Diseño, es un modelo de objetos que describe la realización física de los Casos de Uso, centrándose en como los requisitos funcionales y no funcionales tienen impacto en el sistema a desarrollar. El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción [51].

3.1 Estilos arquitectónicos y patrones de diseño

Tal como se había mencionado Symfony es un *framework* diseñado para agilizar el desarrollo de aplicaciones Web a partir de sus características claves. Separa la lógica de negocio, la lógica del servidor y la presentación de la aplicación. Contiene numerosas herramientas y clases con el objetivo de acortar el tiempo de desarrollo de una aplicación compleja y le permite al desarrollador centrarse en los aspectos específicos de la aplicación al automatizar las tareas comunes. El resultado de estas ventajas es que no se necesita reinventar la rueda cada vez que se construye una nueva aplicación.

3.1.1 Implementación del Modelo Vista Controlador por Symfony

Symfony ha sido escrito completamente en PHP versión 5 con el objetivo de aprovechar todas las ventajas de este lenguaje y se basa en el patrón arquitectónico Modelo Vista Controlador (MVC), implementándolo de modo que el desarrollo de aplicaciones sea rápido y sencillo. Además, se hace una separación en capas más allá del MVC tal como se muestra en la siguiente figura.

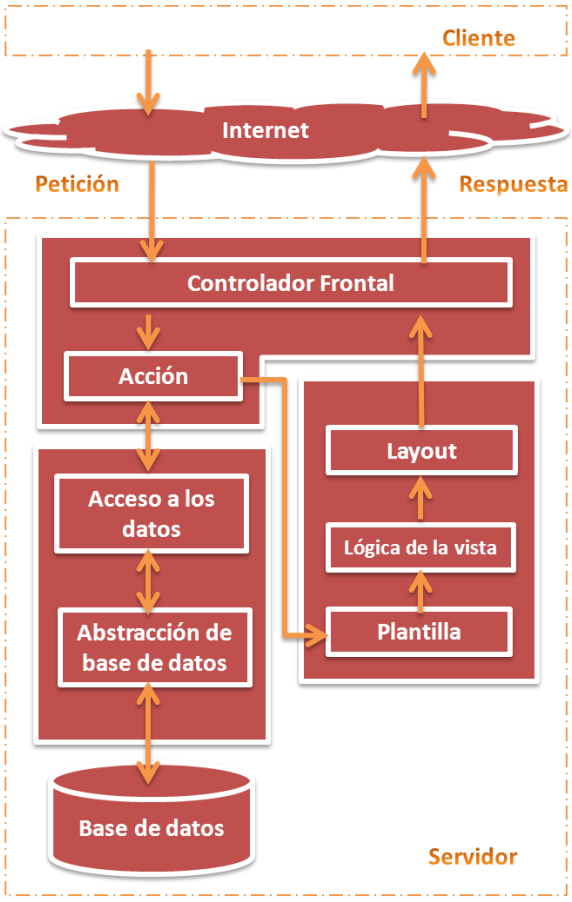


Figura 4: Esquema del Modelo Vista Controlador en Symfony.

En esta variante el controlador que contiene el código que une la lógica de negocio con la presentación, está compuesto por el controlador frontal que es el único punto de entrada a la aplicación y las acciones que se encargan de verificar la validez de las peticiones y preparar los datos para la vista. Esta última está formada por el *layout* que posee el código común a todas las acciones, las plantillas que contienen el código asociado a cada acción en particular y la lógica que se puede manipular a través de un fichero de configuración sencillo sin necesidad de programarla. Por último, el modelo está formado por la capa de acceso a datos cuyas clases son generadas automáticamente por Propel (ORM) en función de las estructuras de datos de la aplicación. Así si se cambia el sistema gestor de bases de datos en cualquier momento, no se debe reescribir ni una línea de código, siendo necesario solamente modificar un parámetro en un archivo de configuración [52].

Tareas del controlador frontal

El controlador frontal es el encargado de gestionar las peticiones de los usuarios, pero ello implica algo más que detectar cual es la acción que se ejecuta. A continuación se muestran cuales son las tareas que realiza:

1. Define las constantes del núcleo.
2. Localiza las bibliotecas de Symfony.
3. Carga e inicializa las clases del núcleo del *framework*.
4. Carga la configuración.
5. Decodifica la URL de la petición para determinar la acción a ejecutar y los parámetros de la petición.
6. Si la acción no existe, redirecciona a la acción error 404.
7. Activa los filtros.
8. Ejecuta los filtros, primera pasada.
9. Ejecuta la acción y produce la vista.
10. Ejecuta los filtros, segunda pasada.
11. Muestra la respuesta.

Filtros

Los filtros son un mecanismo que Symfony usa para realizar las tareas que son comunes a toda la aplicación, tales como la validación y la seguridad. De hecho, cada petición se procesa como una cadena de filtros que son ejecutados de forma sucesiva. A continuación se mencionan cuales son los filtros que se utilizan en Smart Keeper [52].

1. sfRenderingFilter: encargado de renderizar la vista.
2. sfBasicSecurityFilter: chequea la seguridad de cada petición.
3. sfCacheFilter: controla el mecanismo de *cache* del *framework*.

4. sfCommonFilter: añade los ficheros Javascript y CSS a la respuesta.
5. sfFlashFilter: elimina las variables flash de la sesión.
6. sfExecutionFilter: se encarga de validar los parámetros de la petición, la ejecución de la acción y de la vista.

3.1.2 Patrones de diseño

Symfony sigue las mejores prácticas y patrones de diseño para el desarrollo de aplicaciones Web. Seguidamente se mencionan cuales son los patrones de diseño más significativos que se utilizan. Las clases que conforman el núcleo están basadas en el patrón Factory Method permitiendo la creación fácil de los objetos y su extensión de forma sencilla.

El patrón Decorator es usado para decorar el *layout* de la aplicación con el *template* asociado a cada acción, dando como resultado la vista que es mostrada al usuario. Esto se ilustra en la siguiente figura.

Figura 5: Diagrama Patrón *decorator* entre el *layout* y el *template*.

El patrón Singleton es usado en Symfony para permitir el acceso desde cualquier lugar de la aplicación a los objetos relacionado con el núcleo del *framework*.

Los patrones anteriores pertenecen a los conocidos como GoF[20]. Además de ellos se utilizan otros descritos por Martin Fowler [53], entre los que están Front controller, Row Data Gateway y Table Data Gateway.

[20] Gan of Four. Patrones de diseño orientado a objetos.

Los dos últimos son usados en las clases de acceso a datos y permiten realizar operaciones sobre un registro de una tabla y un conjunto de estos respectivamente.

Validación y tratamiento de errores

La validación y el tratamiento de errores en la aplicación se han diseñado en torno a los mecanismos que Symfony provee.

Posee un mecanismo de validación en el servidor a través del uso de ficheros de validación y una serie de validadores que permiten hacer este proceso más fácil y extensible, sin la consecuente promoción de errores que puede traer consigo la programación relacionada con este aspecto. No obstante, hay casos especiales donde es necesario hacer la validación manualmente o combinar ambos métodos. Una característica importante de Symfony en este sentido es que permite el relleno automático de datos (*repopulation*), lo que asegura la obtención de datos correctos y mejora la experiencia de los usuarios. Tal como se mencionó anteriormente la validación se realiza como parte de la ejecución del filtro sfExecutionFilter.

Para el tratamiento de excepciones se utilizará el mecanismo provisto por PHP y las clases que para este fin Symfony posee.

3.1.3 Seguridad

Autenticación y autorización.

Para el diseño del módulo es importante tener en cuenta la seguridad. En este contexto es importante prestar atención a la autenticación y autorización de los usuarios. La interfaz actual de Smart Keeper se vale de Symfony para proveer todo lo necesario para que ambas cosas puedan realizarse de forma sencilla, mejorando el mecanismo de sesiones que utiliza PHP y haciéndolo más configurable y fácil de usar. Una vez que un usuario se autentica, se le asignan los privilegios necesarios (credenciales) para que pueda utilizar las funcionalidades que brinda la aplicación según el rol que desempeñe.

Defensa contra ataques

Existen numerosos ataques informáticos que pueden afectar a las aplicaciones web. Cuando se produce alguno de ellos es porque se explota alguna vulnerabilidad existente relacionada con los principios de

diseño e implementación y con los mecanismos de control o defensa con que se cuenten. A continuación se describen brevemente los principales ataques que podrían afectar la seguridad de la aplicación y por tanto la integridad de los datos que se manejan, así como los principios que regirán la prevención de los mismos en el contexto de Symfony [54].

Cross Site Scripting (XSS) es un tipo de ataque que aprovecha la confianza que un usuario posee en un sitio determinado, se basa en la inyección de códigos hechos en Javascript que pueden impedir el uso normal de una aplicación. Para su prevención Symfony posee un mecanismo de escape que puede ser activado mediante un parámetro en un fichero de configuración [54].

SQL *injection* es un tipo de ataque que permite realizar acciones con el objetivo de corromper la información de un sistema o ganar acceso al mismo a través de la entrada de sentencias SQL maliciosas por medio de formularios o como parámetros de la petición en la URL. EL uso de Propel y Creole ayuda a prevenirlo, ya que ambos poseen los mecanismos de escape necesarios para este tipo de ataque [54].

3.2 Diagramas de clases del diseño

A continuación se muestran los diagramas de clases del diseño para el caso de uso Gestionar Servicios específicamente para el Servicio Squid. Los diagramas para los demás casos de uso así como la descripción de las principales clases pueden ser consultados en el documento anexo “Modelo de diseño”.

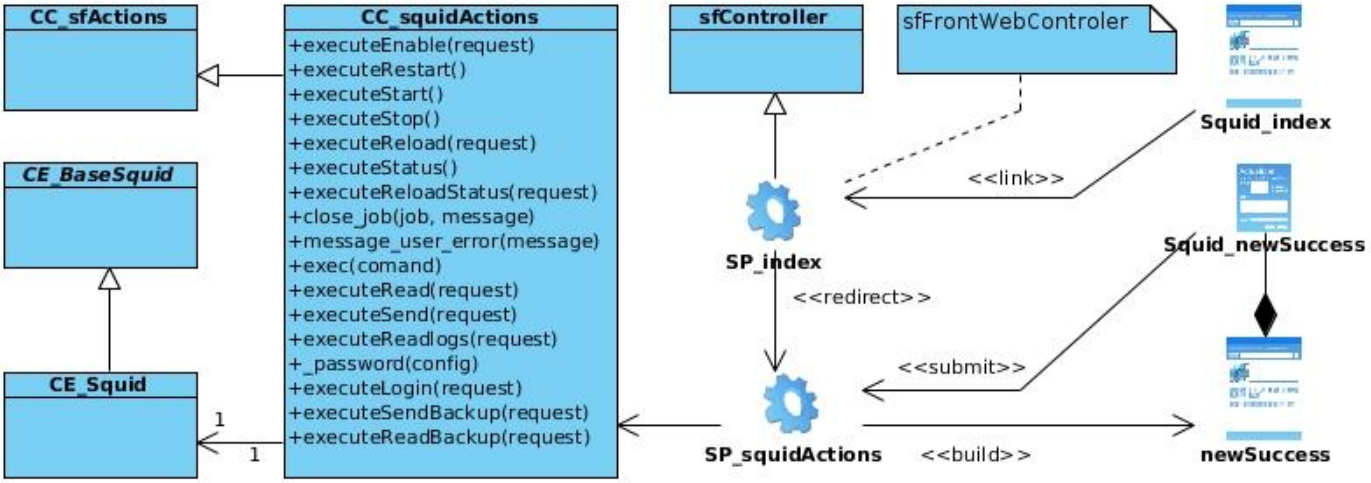


Figura 6: Diagrama de clases del diseño CU Gestionar Servicios - Squid - Sección Adicionar

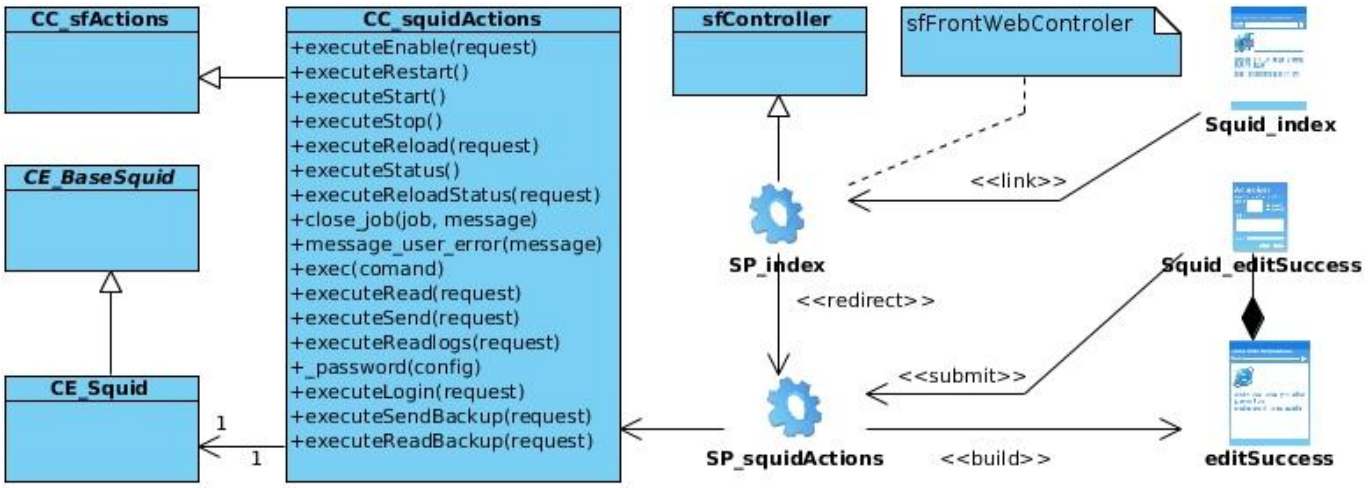


Figura 7: Diagrama de clases del diseño CU Gestionar Servicios - Squid - Sección Editar

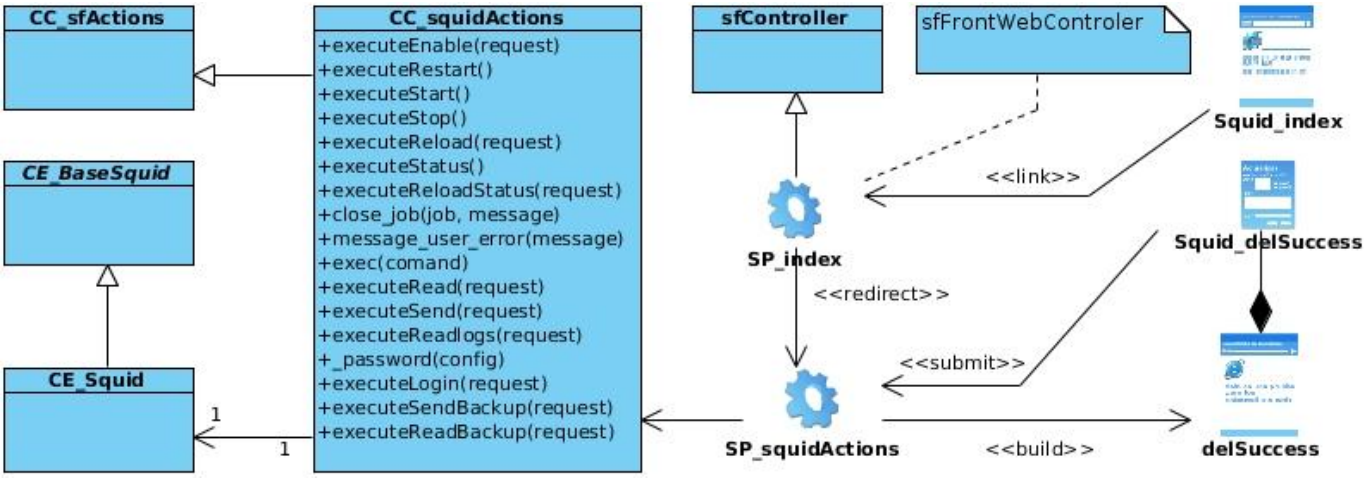


Figura 8: Diagrama de clases del diseño CU Gestionar Servicios - Squid - Sección Eliminar

3.2 Diagrama de secuencia

Los diagramas de secuencia describen como interactúan los objetos del diseño, estos contienen las instancias de los actores, los objetos del diseño y las transmisiones de mensajes entre estos [55].

A continuación se muestran los diagramas de secuencia para el caso de uso Gestionar Servicios, específicamente para el servicio Squid. Los diagramas para los demás casos de uso, así como la

descripción de cada una de las clases que los componen pueden ser consultados en el documento anexo “Modelo de diseño”.

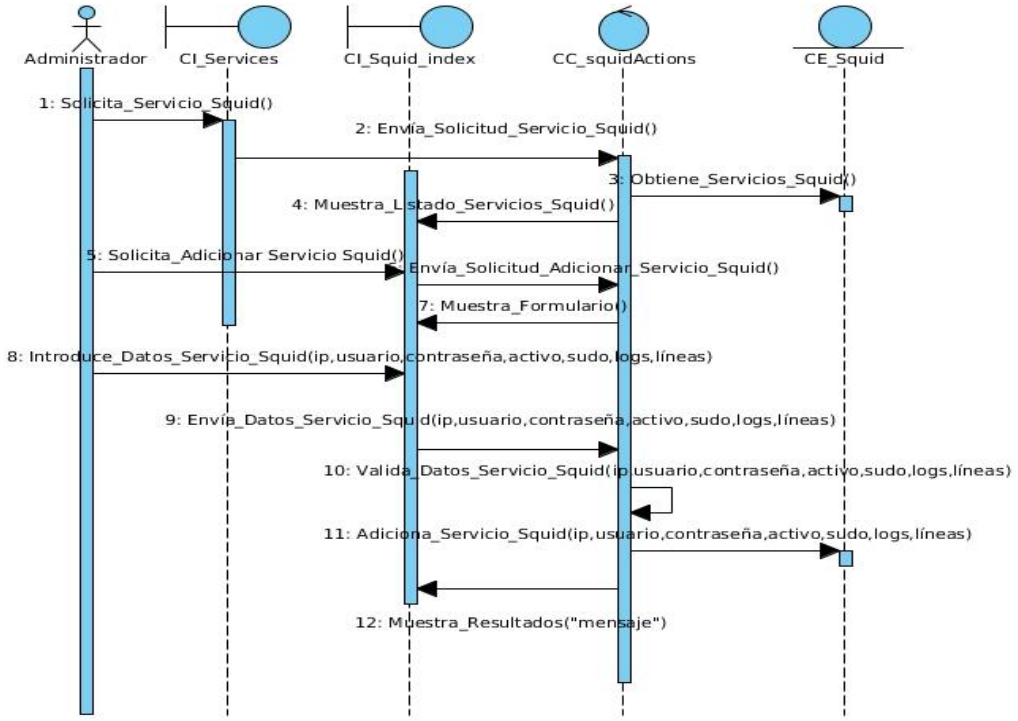


Figura 9: Diagrama de secuencia CU Gestionar Servicios - Squid - Sección Adicionar

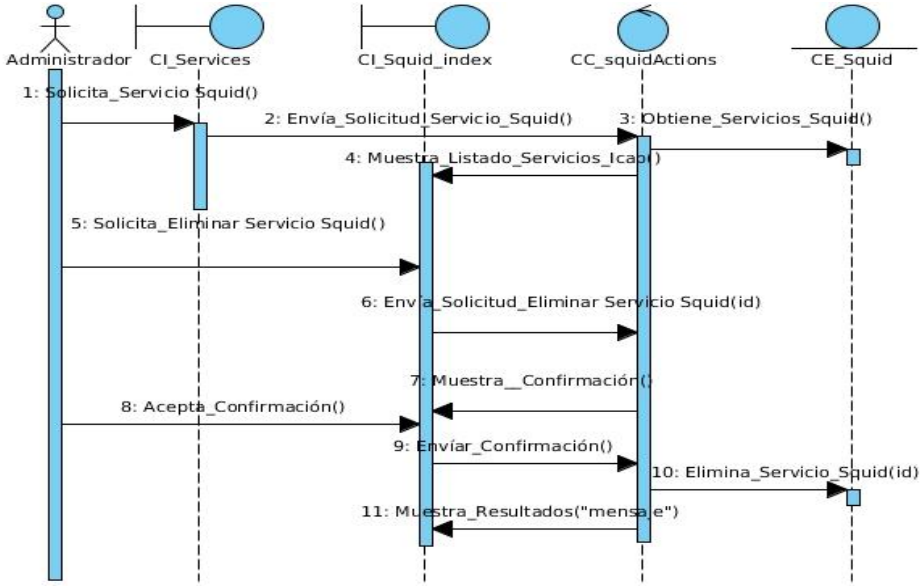


Figura 10: Diagrama de secuencia CU Gestionar Servicios - Squid - Sección Eliminar

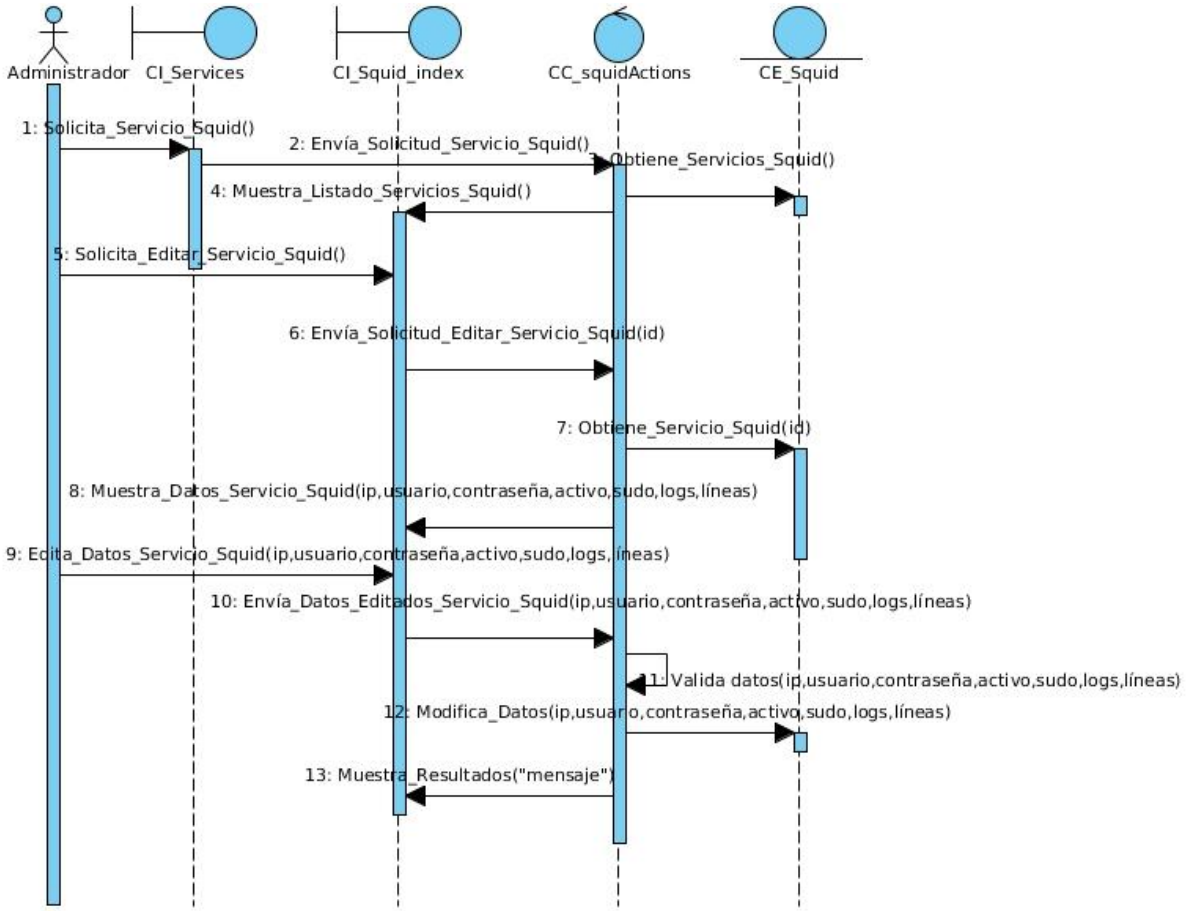


Figura 11: Diagrama de secuencia CU Gestionar Servicios - Squid - Sección Editar

3.3 Diagrama de despliegue

El diagrama de Despliegue muestra las relaciones físicas entre los componentes de *hardware* y *software* en el sistema final. Es un grafo de nodos unidos por conexiones de comunicación donde cada nodo puede contener instancias de componentes. En general un nodo puede ser una unidad de computación de algún tipo, desde un sensor hasta un mainframe y las instancias de componentes de *software* pueden estar unidas por relaciones de dependencia [55].

A continuación se describen cada uno de los nodos presentes en el siguiente diagrama de despliegue y la comunicación entre ellos.

PC Cliente: Representa las computadoras clientes que se conectan al servidor de aplicaciones, la misma se comunica con el servidor mediante el protocolo seguro HTTPS.

Servidor Web: Representa el servidor donde se encuentra instalada la aplicación WEB. Este accede al servidor de Base Datos para el manejo de la información mediante el protocolo TCP/IP.

Servidor de base de datos: Es donde se almacena todos los datos pertenecientes a la aplicación.

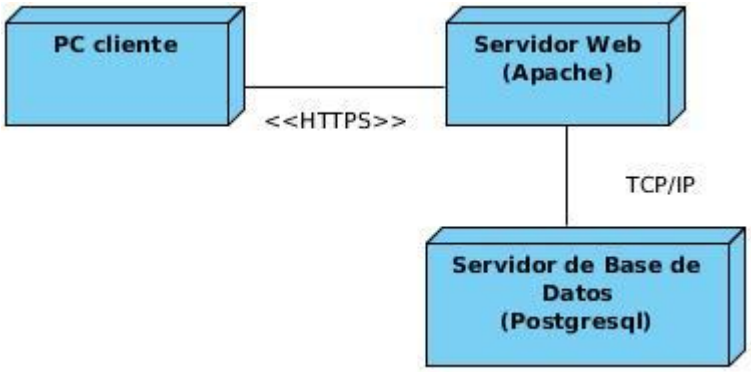


Figura 12: Diagrama de despliegue

3.4 Modelo de datos

La información persistente en la base de datos relacionada con cada uno de los servicios: Apache, Clamav, ICAP, Redirector, Squid y PostgreSQL se almacenará en las tablas de igual nombre respectivamente. Las tablas apache_configs, clamav_configs, icap_configs, redir_configs, squid_configs, y postgres_configs almacenarán todas las direcciones de los ficheros de configuración para cada uno de los servicios correspondientes. El diagrama Entidad-Relación, puede ser consultado en el Anexo C.

3.4.1 Tablas del modelo de datos

A continuación se describen las tablas del modelo de datos del servicio Squid y sus atributos, las tablas correspondientes a los demás servicios se muestran en el Anexo D:

Nombre: squid		
Descripción: Contiene la información de cada uno de los servidores squid.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.

ip	varchar(255)	Almacena la dirección IP donde se encuentra el servicio.
username	varchar(255)	Almacena el nombre del usuario con acceso de administración para acceder a la configuración del servicio.
password	varchar(255)	Almacena la contraseña del usuario con acceso de administración para acceder a la configuración del servicio.
pathlogs	varchar(255)	Almacena la dirección de los *logs* de cada uno de los servicios.
status	bit	Almacena el estado del servicio.
line_to_read	integer	Almacena la cantidad de líneas a leer de los ficheros de *logs.*
sudo	bit	Almacena si es o no necesario utilizar el comando sudo para realizar las acciones de configuración.

Tabla 9: Tabla del modelo de datos squid

Nombre: squid _configs		
Descripción: Contiene la dirección de todos los ficheros de configuración de cada uno de los servicios Squid.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
idf	varchar(255)	Llave foránea de la tabla squid.
config	varchar(255)	Almacena la dirección de todos los ficheros de configuración de cada uno de los servicios.

Tabla 10: Tabla del modelo de datos squid_configs

3.5 Conclusiones

La generación de los artefactos relacionados con el flujo de análisis y diseño, teniendo en cuenta la arquitectura MVC que Symfony establece, permitió obtener una mayor comprensión de la aplicación y definir los principios que guiarán la implementación y organización de la misma. A través del modelado del análisis se obtuvo una visión general conceptual del sistema y mediante el modelado del diseño se obtuvo una abstracción de la implementación del sistema.

Al concluir el presente capítulo se han creado las condiciones para efectuar la implementación del módulo para la gestión remota de los servicios que componen a Smart Keeper.

Capítulo 4

Implementación y pruebas

Introducción

El flujo de implementación y prueba es el más trascendente de la fase de construcción en RUP. Como resultado del mismo, deben traducirse los modelos generados en las fases previas en componentes que formen la aplicación final, reflejando cuales son, su relación y como habrán de distribuirse en los nodos físicos a través de nuevos diagramas. La realización de pruebas también adquiere alta importancia pues permite vislumbrar la existencia de defectos en la implementación realizada [55].

4.1 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de *software*, sean estos ficheros de código fuente, binarios o ejecutables. Representa la estructura del sistema en términos de implementación a un alto nivel. Los elementos de modelado que lo conforman son los componentes y paquetes [55].

A continuación se muestra el diagrama global de paquetes donde se muestran los submódulos que componen al módulo final.

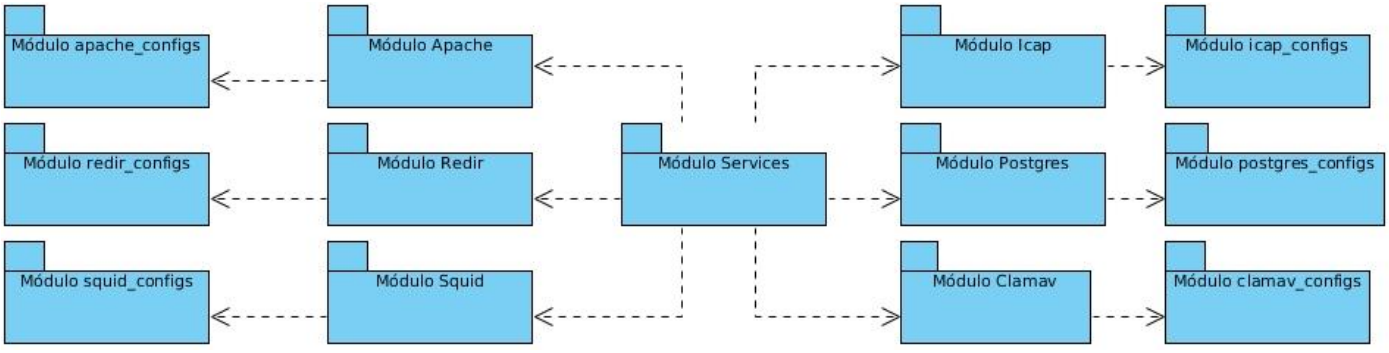


Figura 13: Diagrama global de paquetes

A continuación se muestra el diagrama de componentes para el módulo Squid. Los demás diagramas pueden ser consultados en el Anexo E.

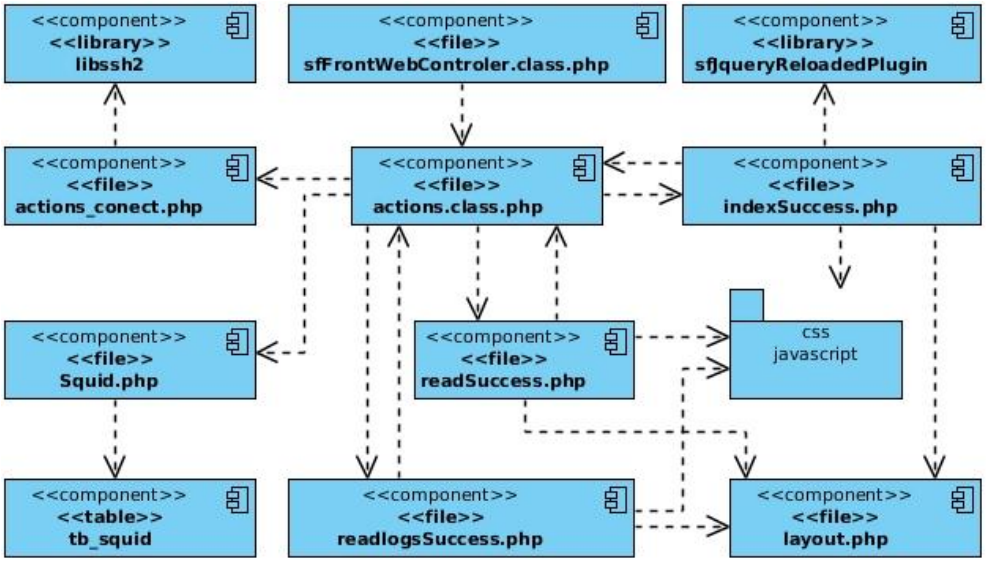


Figura 14: Diagrama de componentes - Módulo Squid

4.2 Código fuente

En el capítulo 1 se seleccionó la biblioteca ***libssh2-php*** del lenguaje de programación PHP para establecer conexiones remotas seguras mediante el protocolo SSH. Para el uso de esta biblioteca se implementó la clase ***actions_conect***. Esta contiene todas las funcionalidades para la realización de las conexiones a los servicios de forma remota. Por su aporte a la investigación a continuación se describen sus principales métodos, los demás pueden ser consultados en el Anexo F.

Método	Descripción	Parámetros de entrada	Retorna
login	Crea una conexión a un servidor remoto a través del protocolo SSH.	**ip**: Dirección del servidor. **usuario**: Usuario de acceso al servidor. **password**: Contraseña de acceso para el usuario en el servidor.	Verdadero en caso de poder realizar la conexión, falso en el caso contrario.

execute_comand	Ejecuta un comando en el servidor remoto.	**cmd**: Comando a ejecutar.	Retorna una cadena de texto con la salida del comando ejecutado.
execute_comand_array	Ejecuta un comando en el servidor remoto.	**cmd**: Comando a ejecutar.	Retorna un arreglo con la salida del comando ejecutado.
read_config	Lee un fichero de configuración pasado por parámetro y verifica si existe.	**config**: Fichero de configuración a leer.	Retorna el fichero de configuración volcado en un arreglo.
send_config_sudo	Guarda un fichero de configuración utilizando sudo en un servidor remoto.	**file**: Dirección del fichero de configuración a salvar en el servidor. **data**: Arreglo con toda la información a adicionar al fichero de configuración. **password**: Contraseña de acceso para utilizar sudo.	
send_config	Guarda un fichero de configuración sin necesidad de utilizar sudo.	**file**: Dirección del fichero de configuración a salvar en el servidor. **data**: Arreglo con toda la información a adicionar al fichero de configuración.	

Tabla 11: Descripción de la clase *actions_conect*

4.3 Vistas principales de la aplicación

A continuación se muestran algunas vistas del módulo desarrollado ya integrado al sistema Smart Keeper. En ellas se puede apreciar que se siguieron las pautas de diseño del sistema original.

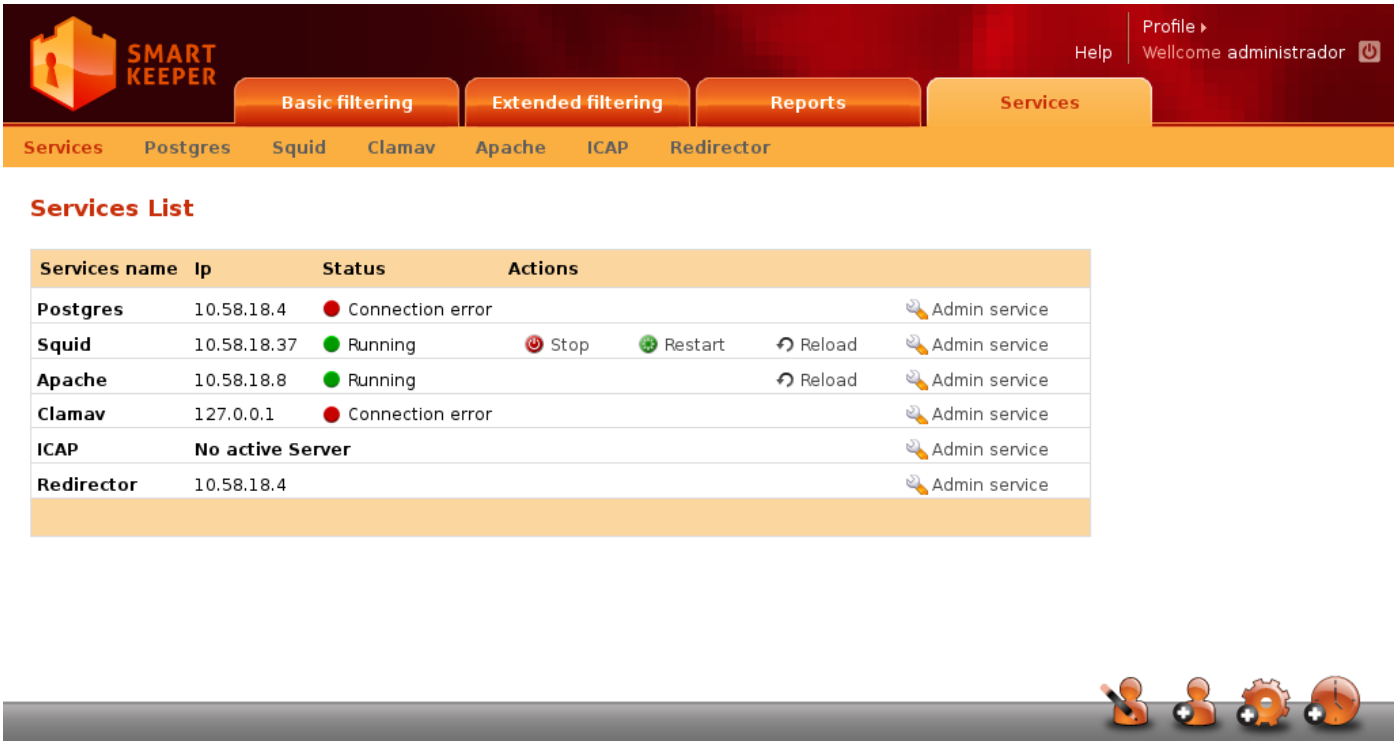


Figura 15: Pantalla del submódulo Services.

Figura 16: Pantalla del submódulo Squid.

4.4 Validación del sistema

Se considera que la prueba de *software* es una etapa imprescindible durante todo el proceso de desarrollo, pues una vez que se genera código fuente, el *software* debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente. Según Pressman[21] el objetivo de la prueba es: "diseñar pruebas que saquen a la luz diferentes clases de errores con la menor cantidad de tiempo y espacio" [55].

4.4.1 Plan de pruebas

Cada prueba que se realice debe ser planificada con antelación, para lograr que los resultados alcanzados sean los esperados y evitar la improvisación durante su ejecución. Antes de realizar las pruebas fueron

[21] Roger S. Pressman. Ingeniero de Software de alto prestigio internacional, fundador y presidente de R.S. Pressman & Associates.

consultados una serie de documentos con el objetivo de tener un pleno conocimiento de las funcionalidades del sistema y todos los requisitos funcionales pertenecientes al sistema fueron profundamente estudiados antes de ponerlos a prueba.

La planificación de las pruebas funcionales quedó reflejada en el documento “Plan de Pruebas”, en el mismo se explica alcance, estrategia, requerimientos a probar y los recursos requeridos, las herramientas así como los responsables involucrados en el proceso de pruebas y constan los entregables como salidas generadas en cada una de las actividades.

4.4.2 Estrategias de Prueba

El método de prueba que se decide aplicar al sistema es el método de Caja Negra, con el fin de estudiar la especificación de las funciones, la entrada y la salida para poder derivar los casos de prueba, definiendo como algo fundamental el probar todas las posibles entradas y salidas del sistema.

4.4.3 Pruebas de caja negra

Las pruebas de caja negra, también denominadas pruebas funcionales o de comportamiento se centran fundamentalmente en los requisitos funcionales. Con el objetivo de facilitar una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes de *software*, se llevan a cabo técnicas de pruebas del *software* como las que se listan a continuación [55]:

- Particiones de equivalencia
- Análisis de valores límites
- Prueba de comparación
- Tabla Ortogonal
- Grafo Causa-efecto

De las técnicas anteriormente mencionadas, se seleccionan para el diseño de los Casos de Pruebas (CP) las técnicas de Partición de equivalencia y Análisis de valores límites. La primera permite simplificar el número de casos de prueba, obteniendo valores válidos e inválidos de las entradas, teniendo como ventaja que cada vez que un usuario usa el programa está a la vez realizando una prueba al mismo. La segunda complementa a la de Partición equivalente porque en lugar de realizar la prueba con cualquier

elemento de la partición equivalente, se escogen los valores en los bordes de la clase. Por lo antes expuesto, es que los autores de este trabajo se disponen a realizar un breve análisis de los métodos seleccionados.

Particiones de equivalencia: Esta técnica utiliza las clases de equivalencia para el diseño de los casos de prueba, es decir, la entrada de una serie de datos válidos y no válidos, cubriendo en cada caso el máximo número de entradas.

Reglas para la identificación de las clases de equivalencia:

- Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

Análisis de valores límites: Esta técnica de prueba complementa a la partición equivalente. En lugar de centrarse solamente en las condiciones de entrada, este obtiene también CP para el campo de salida.

Reglas para el análisis de valores límites:

- Si una condición de entrada especifica un rango delimitado por los valores a y b, se deben diseñar casos de prueba para los valores a y b, y para los valores justo por debajo y justo por encima de a y b, respectivamente.
- Si una condición de entrada especifica un número de valores, se deben desarrollar casos de prueba que ejerciten los valores máximos y mínimos, uno más el máximo y uno menos el mínimo.
- Si las estructuras de datos internas tienen límites preestablecidos hay que asegurarse de diseñar un CP que ejercite la estructura de datos en sus límites.

4.4.4 Diseño de las pruebas

Para realizar el diseño de los casos de prueba funcionales se debe contar con la Especificación de caso de uso y el sistema. El diseño de las pruebas y los resultados de las mismas, quedaron establecidos en los documentos “Diseño de Casos de Pruebas funcionales”, elaborados para cada uno de los requisitos funcionales.

Los requisitos funcionales se dividieron en secciones relacionadas con los servicios a los que se aplican y para cada una de estas se realizó un CP. Estos están formados por las clases de equivalencias diseñadas, algunas válidas y otras no válidas, obteniéndose un resultado en cada una de las pruebas que se ejecutaron. Los CP identificados se relacionan a continuación:

Nombre del caso de prueba	Clases de equivalencias diseñadas
Adicionar servicios.	Adicionar servicio Apache
	Adicionar servicio Clamav
	Adicionar servicio ICAP
	Adicionar servicio PostgreSQL
	Adicionar servicio Redirector
	Adicionar servicio Squid
Eliminar servicios.	Eliminar servicio Apache
	Eliminar servicio Clamav
	Eliminar servicio ICAP
	Eliminar servicio PostgreSQL
	Eliminar servicio Redirector
	Eliminar servicio Squid
Editar servicios.	Editar servicio Apache
	Editar servicio Clamav
	Editar servicio ICAP
	Editar servicio PostgreSQL
	Editar servicio Redirector
	Editar servicio Squid
Adicionar ficheros de configuración.	Adicionar ficheros de configuración a un servicio Apache
	Adicionar ficheros de configuración a un servicio Clamav
	Adicionar ficheros de configuración a un servicio ICAP
	Adicionar ficheros de configuración a un servicio PostgreSQL

	Adicionar ficheros de configuración a un servicio Redirector
	Adicionar ficheros de configuración a un servicio Squid
Eliminar ficheros de configuración.	Eliminar ficheros de configuración a un servicio Apache
	Eliminar ficheros de configuración a un servicio Clamav
	Eliminar ficheros de configuración a un servicio ICAP
	Eliminar ficheros de configuración a un servicio PostgreSQL
	Eliminar ficheros de configuración a un servicio Redirector
	Eliminar ficheros de configuración a un servicio Squid
Editar ficheros de configuración.	Editar ficheros de configuración a un servicio Apache
	Editar ficheros de configuración a un servicio Clamav
	Editar ficheros de configuración a un servicio ICAP
	Editar ficheros de configuración a un servicio PostgreSQL
	Editar ficheros de configuración a un servicio Redirector
	Editar ficheros de configuración a un servicio Squid
Configurar servicios.	Configurar servicio Apache
	Configurar servicio Clamav
	Configurar servicio ICAP
	Configurar servicio PostgreSQL
	Configurar servicio Redirector
	Configurar servicio Squid
Iniciar los servicios.	Iniciar servicio Apache
	.Iniciar servicio Clamav
	Iniciar servicio ICAP
	Iniciar servicio PostgreSQL
	Iniciar servicio Redirector

	Iniciar servicio Squid
Detener los servicios.	Detener servicio Apache
	Detener servicio Clamav
	Detener servicio ICAP
	Detener servicio PostgreSQL
	Detener servicio Redirector
	Detener servicio Squid
Restaurar los servicios.	Restaurar servicio Apache
	Restaurar servicio Clamav
	Restaurar servicio ICAP
	Restaurar servicio PostgreSQL
	Restaurar servicio Redirector
	Restaurar servicio Squid
Reiniciar los servicios.	Reiniciar servicio Apache
	Reiniciar servicio Clamav
	Reiniciar servicio ICAP
	Reiniciar servicio PostgreSQL
	Reiniciar servicio Redirector
	Reiniciar servicio Squid
Mostrar estado de los servicios.	Mostrar estado del servicio Apache
	Mostrar estado del servicio Clamav
	Mostrar estado del servicio ICAP
	Mostrar estado del servicio PostgreSQL
	Mostrar estado del servicio Redirector
	Mostrar estado del servicio Squid
Mostrar los *logs* de los servicios.	Mostrar *logs* del servicio Apache
	Mostrar *logs* del servicio Clamav
	Mostrar *logs* del servicio ICAP

	Mostrar *logs* del servicio PostgreSQL
	Mostrar *logs* del servicio Redirector
	Mostrar *logs* del servicio Squid
Mostrar listado de servicios activos.	Mostrar listado de servicios activos

La siguiente tabla resume la información derivada del diseño y aplicación de las pruebas, para los 79 casos de prueba diseñados se obtuvo un total de 112 clases válidas y 93 no válidas.

Nombre del caso de prueba	Clases de equivalencias diseñadas
Adicionar servicio Squid	72 Clases de equivalencias, 36 válidas y 36 inválidas
Eliminar servicio Squid	12 Clases de equivalencias, 6 válidas y 6 inválidas
Editar servicio Squid	72 Clases de equivalencias, 36 válidas y 36 inválidas
Adicionar ficheros de configuración a un servicio Squid	8 Clases de equivalencias, 4 válidas y 4 inválidas
Eliminar ficheros de configuración a un servicio Squid	6 Clases de equivalencias válidas
Editar ficheros de configuración de un servicio Squid	8 Clases de equivalencias, 4 válidas y 4 inválidas
Configurar servicio Squid	3 Clases de equivalencias válidas
Iniciar servicio Squid	2 Clases de equivalencias válidas
Detener servicio Squid	2 Clases de equivalencias válidas
Restaurar servicio Squid	2 Clases de equivalencias válidas
Reiniciar servicio Squid	2 Clases de equivalencias válidas
Mostrar estado de un servicio Squid	2 Clases de equivalencias válidas
Mostrar *logs* del servicio Squid	6 Clases de equivalencias válidas
Mostrar listado de servicios activos	1 Clase de equivalencia válida

La información completa del diseño de casos de prueba del sistema; puede ser consultada en el documento “Diseño de Casos de pruebas funcionales”.

4.4.5 Pruebas aceptación y de integración

Las pruebas de integración se utilizan para verificar que los componentes interactúen entre sí de la forma apropiada después de haber sido integrados a un sistema [55]. El módulo desarrollado fue integrado al sistema Smart Keeper para comprobar su correcto funcionamiento, además como parte del proceso de pruebas fue instalado en un sistema desplegado en la UCI, que es usado por el Centro de Ideoinformática (CIDI). Estas pruebas servirán para comprobar la aceptación del producto.

4.4.6 Resultados de las Pruebas

Se realizaron tres iteraciones de pruebas funcionales donde se detectaron varias no conformidades. En la primera iteración se encontraron un total de 14 no conformidades, que en la mayoría de los casos estuvieron dadas por problemas de validación y errores ortográficos. En el transcurso de la etapa de pruebas se fueron mitigando las no conformidades detectadas, con el objetivo de corregir los errores encontrados y garantizar la calidad de la aplicación.

El Ing. Luis E. Sánchez Arce líder del proyecto Smart Keeper y encargado de administrar el sistema desplegado en el CIDI, avala la integración del módulo y que este cumple con todos los requisitos necesarios que permiten la fácil administración de los servicios que integran al sistema. Los avales se pueden encontrar en el documento “Aval de aceptación y de integración”.

4.5 Conclusiones

Tras el flujo de implementación y prueba el sistema quedó desarrollado. En los diagramas generados pudo ilustrarse la relación entre los principales componentes del sistema y como estará distribuido este. Las pruebas realizadas arrojaron algunos defectos que fueron corregidos y permitieron aumentar la calidad final de la solución. El módulo desarrollado aprovechó las ventajas de Symfony quedando correctamente integrado al sistema Smart Keeper.

Conclusiones

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

- El análisis de aplicaciones similares efectuado arrojó que existe déficit de aplicaciones de escritorio o web para administrar y configurar servicios instalados en diferentes servidores desde una misma interfaz en los sistemas GNU/Linux.
- El uso del protocolo SSH para la administración remota de servidores, contribuye con la seguridad y la estabilidad de las conexiones a los servidores.
- La metodología RUP se integró perfectamente al desarrollo del módulo aumentando la calidad del producto final.
- Después de haber probado e integrado el módulo al sistema Smart Keeper y teniendo en cuenta las características y funcionalidades del mismo se concluye que facilitará el trabajo de los administradores en el sistema, ratificando la idea a defender.

En resumen, los principales aportes de este trabajo son:

- Base teórica para el desarrollo de sistemas afines.
- Base para el desarrollo de diferentes módulos para administrar y configurar nuevos servicios que se integren al sistema Smart Keeper.

Los elementos planteados anteriormente permiten afirmar que se cumplió el objetivo de desarrollar un módulo que permita de forma remota, administrar y configurar los servicios que componen al sistema Smart Keeper.

Recomendaciones

Después de haber concluido con el desarrollo de una primera versión del módulo para administrar y configurar los servicios que componen al sistema Smart Keeper se recomienda desarrollar en futuras versiones las siguientes funcionalidades:

- Permitir cambiar el estado y configurar los servicios sin necesidad de tener que ponerlos activos.
- Agregar una interfaz para configurar parámetros básicos de los ficheros de configuración.
- Agregar funcionalidad *Ping* para conocer el estado del servidor donde se ejecute un determinado servicio.
- Agregar al módulo PostgreSQL la posibilidad de crear salvas de la base de datos.
- Agregar la posibilidad de autenticación en las conexiones remotas utilizando llave pública.

Referencias bibliográficas

[1] **Machine-Tool Institute.** Herramientas de Control de Accesos y Monitorización. [En línea] 2011 Disponible en: <http://www.imh.es/dokumentazio-irekia/manuales/seguridad-basica-en-internet/filtro-de-contenidos/herramientas-de-control-de-accesos-y-monitorizacion-para-el-pc> [Citado en 2012].

[2] **Guerrero Enamorado, Alain.** Concepción del Sistema de Filtrado para Internet. Universidad de las Ciencias Informáticas, La Habana, 2009, 97 p.

[3] **Universidad de las Ciencias Informáticas.** Smart Keeper. Manual de usuario. La Habana, 2011, 64 p.

[4] **Sánchez Arce, Luis Enrique.** Interfaz de Administración Web para el Sistema de Filtrado Filpacon. Universidad de las Ciencias Informáticas, La Habana, 2008, 190 p.

[5] **Grupo ADSLzone.** Diccionario informático. [En línea] 2012. Disponible en: <http://www.softzone.es/glosario/a-b-y-c/> [Citado en 2012].

[6] **Webmin community.** What is Webmin? [En línea] 1 de diciembre 2011. Disponible en: <http://www.webmin.com> [Citado en 2012].

[7] **Jorba Esteve, Josep y Suppi Boldrito, Remo.** Administración avanzada de GNU/Linux. Barcelona, Eureca Media, 2004, 472 p.

[8] **Comunidad de Ubuntu.** Documentación oficial del proyecto Zential. [En línea] 2012 Disponible en: <http://doc.zentyal.org/es/zindex.html> [Citado en 2012].

[9] Ídem [7].

[10] **Universidad del Saber.** Definición de CONEXIÓN REMOTA. [En línea] 2011 Disponible en: <http://www.hooping.net/glossary/conexion-remota-27.aspx> [Citado en 2012].

[11] **Seifried, Kurt.** Guía de Seguridad del Administrador de Linux. [En línea] 2011. Disponible en: <http://es.tldp.org/Manuales-LuCAS/GSAL/gsal-19991128.pdf> [Citado en 2012].

[12] Ídem [7].

[13] **Kriptópolis.** FreeNX: Acceso remoto al escritorio Linux. [En línea] 2011 Disponible en: <http://www.kriptopolis.org/freenx-acceso-remoto-escritorio-linux> [Citado en 2012].

[14] Ídem [11].

[15] **OpenSSH**. Documentación Oficial del Proyecto. [En línea] 2012 Disponible en: http://www.openssh.org/es/features.html [Citado en 2012].

[16] **Johnston, Matt.** Dropbear SSH server and client. [En línea] 2011 Disponible en: <http://matt.ucc.asn.au/dropbear/dropbear.html> [Citado en 2012].

[17] **PuTTY.ORG.** PuTTY. [En línea] 2012 Disponible en: <http://www.putty.org> [Citado en 2012].

[18] **Vinagre.** Vinagre [En línea] 2012 Disponible en: <http://projects.gnome.org/vinagre> [Citado en 2012].

[19] **Gnome Documentation Group.** Vino. [En línea] 2012 Disponible en: <http://live.gnome.org/Vino> [Citado en 2012].

[20] **Chao, Thomas**. Linux XDMCP HOWTO. [En línea] 2008 Disponible en: <http://www.tldp.org/HOWTO/XDMCP-HOWTO/index.html> [Citado en 2011].

[21] Ídem [7].

[22] Ídem [11].

[23] **Barrett, Daniel J.** SSH. The secure Shell. The definitive guide. O'Reilly Media, Inc., 2011. 668 p.

[24] Ídem [15]

[25] **Creación web gratis.** Ventajas y desventajas del Personal Home Page PHP. [En línea] 2011. Disponible en: <http://www.creargratisunapaginaweb.com/php/ventajas-y-desventajas-del-personal-home-page-4/> [Citado en 2011].

[26] **W3C**. What is HTML? [En línea] 2012. Disponible en: <http://www.w3.org/MarkUp/> [Citado en 2012].

[27] **Free Software Foundation, Inc**. What is Bash? [En línea] 2012. Disponible en: <http://www.gnu.org/software/bash/manual/html_node/What-is-Bash_003f.html> [Citado en 2012].

[28] **PHP Documentation Group.** SSH2 - Manual. [En línea] 2011. Disponible en: <http://php.net/manual/en/book.ssh2.php> [Citado en 2012].

[29] **Larsen, Audun.** A PHP security library. [En línea] 2012. Disponible en: <http://phpseclib.com> [Citado en 2012].

[30] **Haas, Juergen.** What is Sudo? [En línea] 2012. Disponible en: <http://linux.about.com/cs/linux101/g/sudo.htm> [Citado en 2012].

[31] **Oracle Corporation.** Netbeans IDE. [En línea] 2012. Disponible en: <http://netbeans.org/features/php> [Citado en 2012].

[32] **Potencier, Fabien.** Symfony la guía definitiva. [En línea] 2010. Disponible en: <http://www.librosweb.es/symfony> [Citado en 2012].

[33] **Metodologías.** Comparación entre las metodologías de desarrollo. [En línea] 2012 Disponible en: <http://metodologiaxpvsmetodologiarup.blogspot.com> [Citado en 2011].

[34] **Kniberg, Henrik**. Scrum and XP from the Trenches. [En línea] 2011 Disponible en: <http://www.eyrie.org/~eagle/reviews/books/1-4303-2264-0.html> [Citado en 2012].

[35] **Proyectos Agiles.** Qué es Scrum. [En línea] 2011 Disponible en: <http://www.proyectosagiles.org/> [Citado en 2011].

[36] **Peñalver Romero, Gladys Marsi**. SXP, metodología ágil para proyectos de Software libre. Universidad de las Ciencias Informáticas, Ciudad de la Habana, 2009, 94 p.

[37] **IBM.** Rational Unified Process. [En línea] 2011 Disponible en: <http://www.rational.com.ar/herramientas/rup.html> [Citado en 2011].

[38] **Ivar Jacobson, Grady Booch, James Rumbaugh.** El Proceso Unificado de Desarrollo de Software. Madrid, Addison-Wesley, 2000, 463 p.

[39] **The Apache Software Foundation**. Documentación Oficial del Proyecto. [En línea] 2012. Disponible en: <https://httpd.apache.org/ABOUT_APACHE.html> [Citado en 2012].

[40] **Civenetia.** Configuración de Apache. [En línea] 2012. Disponible en: <http://www.cibernetia.com/manuales/instalacion_servidor_web/2_3_configuracion_apache.php> [Citado en 2012].

[41] **PostgreSQL Global Development Group.** About. [En línea] 2011 Disponible en: <http://www.postgresql.org/about> [Citado en 2012].

[42] **LINUX-ES.ORG**. Introducción a PostgreSQL – Configuración. [En línea] 2011 Disponible en:

<http://www.linux-es.org/node/660> [Citado en 2012].

[43] **ClamAV.** About ClamAV. [En línea] 2011 Disponible en: <http://www.clamav.net/lang/en/about> [Citado en 2012].

[44] **Kojm, Tomasz.** User Manual. [En línea] 2011 Disponible en: <http://www.clamav.net/doc/latest/html/> [Citado en 2012].

[45] **Greasyspoon.** Greasyspoon: Fábrica de secuencias de comandos para los servicios básicos de red. [En línea] 2012. Disponible en: <http://greasyspoon.sourceforge.net/index.html> [Citado en 2012].

[46] **Squid.** Optimising Web Delivery [En línea] 2011 Disponible en: <http://www.squid-cache.org/Intro/> [Citado en 2012].

[47] **Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado**. Instalación y Configuración de Squid3, [En línea] 2012 Disponible en: <http://www.ite.educacion.es/formacion/materiales/85/cd/REDES_LINUX/enrutamiento/Proxy_squid.html> [Citado en 2012].

[48] Ídem [3]

[49] **Pressman, R. S.** Software Engineering. A practitioner´s approach. NY, Mc Graw Hill, 2010, 852 p.

[50] **Young, Ralph R**. The Requirements Engineering Handbook. United States of America, ARTECH HOUSE, 2004, 275 p.

[51] **Sommerville, Ian**. Software Engineering, Madrid, Addison-Wesley, 2007, 823 p.

[52] Ídem [32].

[53] **Martin Fowler**. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2012.

[54] Ídem [4].

[55] Ídem [49].

Anexo A

Figuras relacionadas

La figura 1 muestra una de las formas en que puede ser instalado el sistema Smart Keeper. En ella se muestra una instalación donde cada uno de los servicios que componen al sistema, se instalan de forma distribuida.

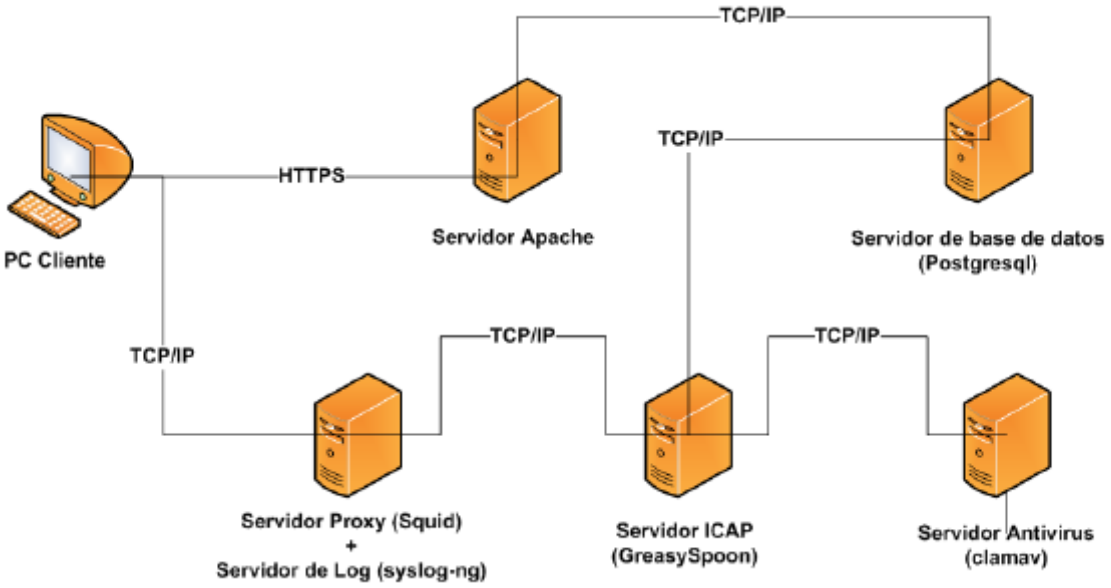


Figura 1: Instalación distribuida

Anexo B

Descripción expandida de los casos de uso del sistema

A continuación se describen cada uno de los casos de uso del sistema.

Caso de Uso	Mostrar Listado de Servicios Activos
Actores	Administrador
Resumen	El caso de uso se inicia cuando el administrador desea ver el listado con todos los servicios activos.
Precondiciones	
Referencias	RF14
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona en el menú principal “Servicios” y se dirige al submenú “Servicios”.	2. El sistema muestra el listado de todos los servicios activos.

Tabla 12: Descripción del caso de uso - Mostrar Listado de Servicios Activos.

Caso de Uso	Gestionar Ficheros de Configuración
Actores	Administrador
Resumen	El caso de uso se inicia cuando el administrador necesita adicionar un fichero de configuración a un servicio.
Precondiciones	Los ficheros de configuración se le adicionan a los servicios por lo que es necesario primeramente haber adicionado y activado un servicio.
Referencias	RF4,RF5,RF6
Prioridad	Crítico
Flujo Normal de Eventos	
Sección “General”	

Acción del Actor	Respuesta del Sistema
1. El administrador selecciona en el menú principal “Servicios” y se dirige al submenú del servicio que desee.	2. El sistema muestra el listado de todos los servicios del tipo seleccionado.
3. El administrador presiona el botón “Adicionar Ficheros”, en el servicio activo.	4. . El sistema muestra el listado de todos ficheros de configuración de dicho servicio.
Sección “Adicionar Fichero de Configuración”	
1. El administrador presiona el botón “Adicionar”	2. El sistema muestra una interfaz donde se muestran los datos requeridos para crear un nuevo fichero de configuración.
3. El administrador introduce los datos y presiona el botón “Guardar”.	4. El sistema valida los datos e inserta el nuevo fichero de configuración. El sistema muestra el mensaje “Elemento creado correctamente.”
Flujos Alternos	
4.1 En caso de ocurrir algún error en los datos entrados el sistema muestra el mensaje “Elemento no guardado, se ha producido algún error.”, inmediatamente se vuelve al paso 4.	
Sección “Eliminar Fichero de Configuración”	
1. El administrador presiona el botón “Borrar”, en la fila del fichero que desee eliminar.	2. El sistema muestra un diálogo para verificar si realmente se desea eliminar el servicio.
3. El administrador presiona el botón “Borrar”	4. El sistema elimina el servicio y muestra el mensaje “Elemento borrado correctamente.”
Flujos Alternos	
3.1 En caso de presionar el botón “Cancelar” no se realiza nada.	
Sección “Editar Fichero de Configuración”	
1. El administrador selecciona la opción “Editar”, en la fila del fichero de configuración que desee editar.	2. El sistema muestra una interfaz donde se muestran todos los datos del fichero de configuración con la posibilidad de ser editados.
3. El administrador edita los datos que desee. 4. Presiona el botón “Guardar”.	5. El sistema valida los datos y guarda el fichero de configuración editado e inmediatamente muestra

	el mensaje “Elemento actualizado correctamente.”.
Flujos Alternos	
5.1 En caso de ocurrir algún error en los datos entrados el sistema muestra el mensaje “Elemento no guardado, se ha producido algún error.”, inmediatamente se vuelve al paso 2.	

Tabla 13: Descripción del caso de uso - Gestionar Ficheros de Configuración.

Caso de Uso	Mostrar *logs* de los Servicios
Actores	Administrador
Resumen	El caso de uso se inicia cuando el administrador desea visualizar los *logs* de un determinado servicio.
Precondiciones	Es necesario tener un servicio activo
Referencias	RF13
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona en el menú principal “Servicios” y se dirige al submenú del servicio que desee.	2. El sistema muestra el listado de todos los servicios del tipo seleccionado.
3. El administrador selecciona el botón principal “*Logs*”.	4. El sistema muestra los logs del servicio activo.
Flujos Alternos	
1. En caso de ocurrir un error en la conexión el sistema muestra el mensaje de error “Error en la conexión”, e inmediatamente se vuelve al paso 2. 2. En caso de que no exista el fichero de *logs* o esté vacío el sistema muestra el mensaje de error “El fichero de *logs* no existe o está vacío”, e inmediatamente se vuelve al paso 2.	

Tabla 14: Descripción del caso de uso - Mostrar *logs* de los Servicios.

Caso de Uso	Configurar Servicios
Actores	Administrador

Resumen	El caso de uso se inicia cuando el administrador desea configurar un determinado servicio.
Precondiciones	Es necesario tener un servicio activo
Referencias	RF7
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona en el menú principal “Servicios” y se dirige al submenú del servicio que desee.	2. El sistema muestra el listado de todos los servicios del tipo seleccionado.
3. El administrador selecciona el fichero de configuración y después presiona el botón “Configurar”, en el servicio activo.	4. El sistema muestra el fichero de configuración.
5. El administrador modifica el fichero de configuración y presiona el botón guardar.	6. El sistema guarda el fichero de configuración en el servidor y luego envía el mensaje “Su configuración ha sido guardada”
Flujos Alternos	
1. En caso de ocurrir un error en la conexión el sistema muestra el mensaje de error “Error en la conexión”, e inmediatamente se vuelve al paso 2. 2. En caso de que no exista el fichero de configuración el sistema muestra el mensaje de error “El fichero de configuración no existe”, e inmediatamente se vuelve al paso 2. 3. En caso de que el usuario no tenga permisos de administración el sistema muestra el mensaje de error “El usuario no es administrador”, e inmediatamente se vuelve al paso 2.	

Tabla 15: Descripción del caso de uso - Configurar Servicios.

Caso de Uso	Cambiar estado de los Servicios
Actores	Administrador
Resumen	El caso de uso se inicia cuando el administrador desea configurar un determinado servicio.

Precondiciones	Es necesario tener un servicio activo
Referencias	RF8, RF9, RF10, RF11, RF12
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona en el menú principal “Servicios” y se dirige al submenú del servicio que desee.	2. El sistema muestra el listado de todos los servicios del tipo seleccionado.
3. El administrador presiona la acción que desea realizar ya sea “Iniciar”, “Detener”, “Reiniciar”, “Restaurar” o “Mostrar Estado”.	4. El sistema ejecuta la acción en el servidor remoto y envía el mensaje devuelto por el servidor.

Flujos Alternos

1. En caso de ocurrir un error en la conexión el sistema muestra el mensaje de error “Error en la conexión”, e inmediatamente se vuelve al paso 2.
2. En caso de que no se encuentre instalado el servicio en el servidor el sistema muestra el mensaje de error “El servicio no se encuentra instalado en el servidor”, e inmediatamente se vuelve al paso 2.
3. En caso de que el usuario no tenga permisos de administración el sistema muestra el mensaje de error “El usuario no es administrador”, e inmediatamente se vuelve al paso 2.

Tabla 16: Descripción del caso de uso - Cambiar estado de los Servicios.

Anexo C

Modelo de datos

A continuación se muestra el modelo de datos utilizado para almacenar toda la información referente a los servicios que componen al sistema Smart Keeper.

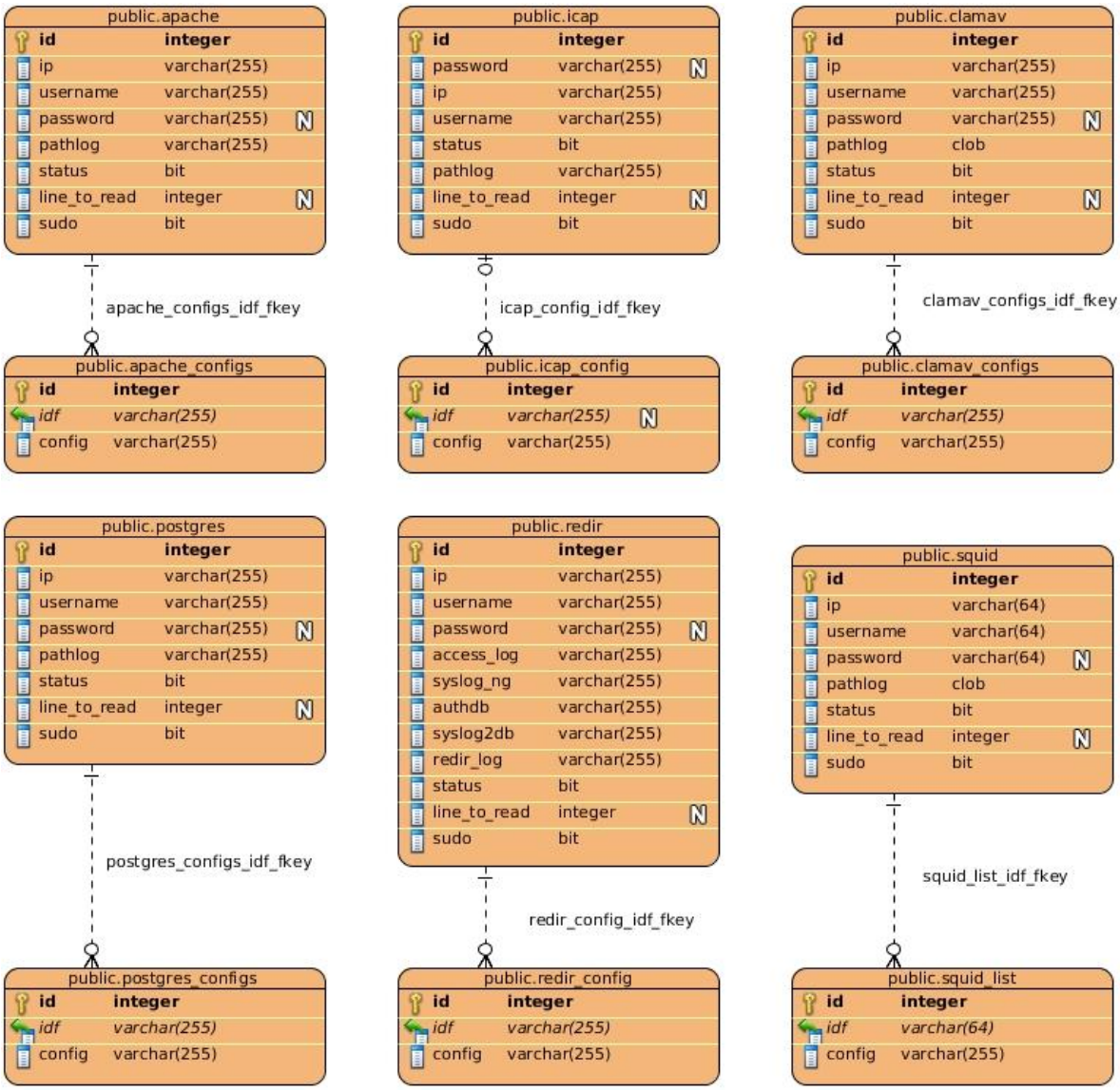


Figura 17: Modelo de datos.

Anexo D

Tablas del modelo de datos

A continuación se describen las tablas del modelo de datos, así como todos sus atributos.

Nombre: apache		
Descripción: Contiene la información de cada uno de los servidores Apache.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
ip	varchar(255)	Almacena la dirección IP donde se encuentra el servicio.
username	varchar(255)	Almacena el nombre del usuario con acceso de administración para acceder a la configuración del servicio.
password	varchar(255)	Almacena la contraseña del usuario con acceso de administración para acceder a la configuración del servicio.
pathlogs	varchar(255)	Almacena la dirección de los *logs* de cada uno de los servicios.
status	bit	Almacena el estado del servicio.
line_to_read	integer	Almacena la cantidad de líneas a leer de los ficheros de *logs*
sudo	bit	Almacena si es o no necesario utilizar el comando sudo para realizar las acciones de configuración.

Tabla 17: Tabla del modelo de datos apache

Nombre: clamav		
Descripción: Contiene la información de cada uno de los servidores clamav.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
ip	varchar(255)	Almacena la dirección IP donde se encuentra el servicio.
username	varchar(255)	Almacena el nombre del usuario con acceso de administración para acceder a la configuración del servicio.
password	varchar(255)	Almacena la contraseña del usuario con acceso de

		administración para acceder a la configuración del servicio.
pathlogs	varchar(255)	Almacena la dirección de los *logs* de cada uno de los servicios.
status	bit	Almacena el estado del servicio.
line_to_read	integer	Almacena la cantidad de líneas a leer de los ficheros de *logs.*
sudo	bit	Almacena si es o no necesario utilizar el comando sudo para realizar las acciones de configuración.

Tabla 18: Tabla del modelo de datos clamav

Nombre: icap		
Descripción: Contiene la información de cada uno de los servidores Greasyspoon.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
ip	varchar(255)	Almacena la dirección IP donde se encuentra el servicio.
username	varchar(255)	Almacena el nombre del usuario con acceso de administración para acceder a la configuración del servicio.
password	varchar(255)	Almacena la contraseña del usuario con acceso de administración para acceder a la configuración del servicio.
pathlogs	varchar(255)	Almacena la dirección de los *logs* de cada uno de los servicios.
status	bit	Almacena el estado del servicio.
line_to_read	integer	Almacena la cantidad de líneas a leer de los ficheros de *logs.*
sudo	bit	Almacena si es o no necesario utilizar el comando sudo para realizar las acciones de configuración.

Tabla 19: Tabla del modelo de datos icap

Nombre: postgres		
Descripción: Contiene la información de cada uno de los servidores postgresql.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
ip	varchar(255)	Almacena la dirección IP donde se encuentra el servicio.
username	varchar(255)	Almacena el nombre del usuario con acceso de administración

		para acceder a la configuración del servicio.
password	varchar(255)	Almacena la contraseña del usuario con acceso de administración para acceder a la configuración del servicio.
pathlogs	varchar(255)	Almacena la dirección de los *logs* de cada uno de los servicios.
status	bit	Almacena el estado del servicio.
line_to_read	integer	Almacena la cantidad de líneas a leer de los ficheros de *logs.*
sudo	bit	Almacena si es o no necesario utilizar el comando sudo para realizar las acciones de configuración.

Tabla 20: Tabla del modelo de datos postgres

Nombre: redir		
Descripción: Contiene la información de cada uno de los servidores redir.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
ip	varchar(255)	Almacena la dirección IP donde se encuentra el servicio.
username	varchar(255)	Almacena el nombre del usuario con acceso de administración para acceder a la configuración del servicio.
password	varchar(255)	Almacena la contraseña del usuario con acceso de administración para acceder a la configuración del servicio.
access_log	varchar(255)	Almacena la dirección de los ficheros de access_log de cada uno de los servicios.
syslog_ng	varchar(255)	Almacena la dirección de los ficheros de syslog_ng de cada uno de los servicios.
authdb	varchar(255)	Almacena la dirección de los ficheros de authdb de cada uno de los servicios.
syslog2db	varchar(255)	Almacena la dirección de los ficheros de syslog2db de cada uno de los servicios.
redir_log	varchar(255)	Almacena la dirección de los ficheros de redir_log de cada uno de los servicios.
status	bit	Almacena el estado del servicio.

line_to_read	integer	Almacena la cantidad de líneas a leer de los ficheros de *logs.*
sudo	bit	Almacena si es o no necesario utilizar el comando sudo para realizar las acciones de configuración.

Tabla 21: Tabla del modelo de datos redir

Nombre: apache_configs		
Descripción: Contiene la dirección de todos los ficheros de configuración de cada uno de los servicios Apache.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
idf	varchar(255)	Llave foránea de la tabla apache.
config	varchar(255)	Almacena la dirección de todos los ficheros de configuración de cada uno de los servicios.

Tabla 22: Tabla del modelo de datos apache_configs

Nombre: clamav_configs		
Descripción: Contiene la dirección de todos los ficheros de configuración de cada uno de los servicios Clamav.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
idf	varchar(255)	Llave foránea de la tabla clamav.
config	varchar(255)	Almacena la dirección de todos los ficheros de configuración de cada uno de los servicios.

Tabla 23: Tabla del modelo de datos clamav_configs

Nombre: icap_configs		
Descripción: Contiene la dirección de todos los ficheros de configuración de cada uno de los servicios ICAP.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
idf	varchar(255)	Llave foránea de la tabla icap.

config	varchar(255)	Almacena la dirección de todos los ficheros de configuración de cada uno de los servicios.

Tabla 24: Tabla del modelo de datos icap_configs

Nombre: postgres_configs		
Descripción: Contiene la dirección de todos los ficheros de configuración de cada uno de los servicios PostgreSQL.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
idf	varchar(255)	Llave foránea de la tabla postgres.
config	varchar(255)	Almacena la dirección de todos los ficheros de configuración de cada uno de los servicios.

Tabla 25: Tabla del modelo de datos postgres_configs

Nombre: redir_configs		
Descripción: Contiene la dirección de todos los ficheros de configuración de cada uno de los servicios Redirector.		
Atributo	**Tipo**	**Descripción**
id	integer	Campo autoincremental.
idf	varchar(255)	Llave foránea de la tabla redir.
config	varchar(255)	Almacena la dirección de todos los ficheros de configuración de cada uno de los servicios.

Tabla 26: Tabla del modelo de datos redir_configs

Anexo E

Diagramas de Componentes

A continuación se muestran los diagramas de componentes de los módulos del sistema.

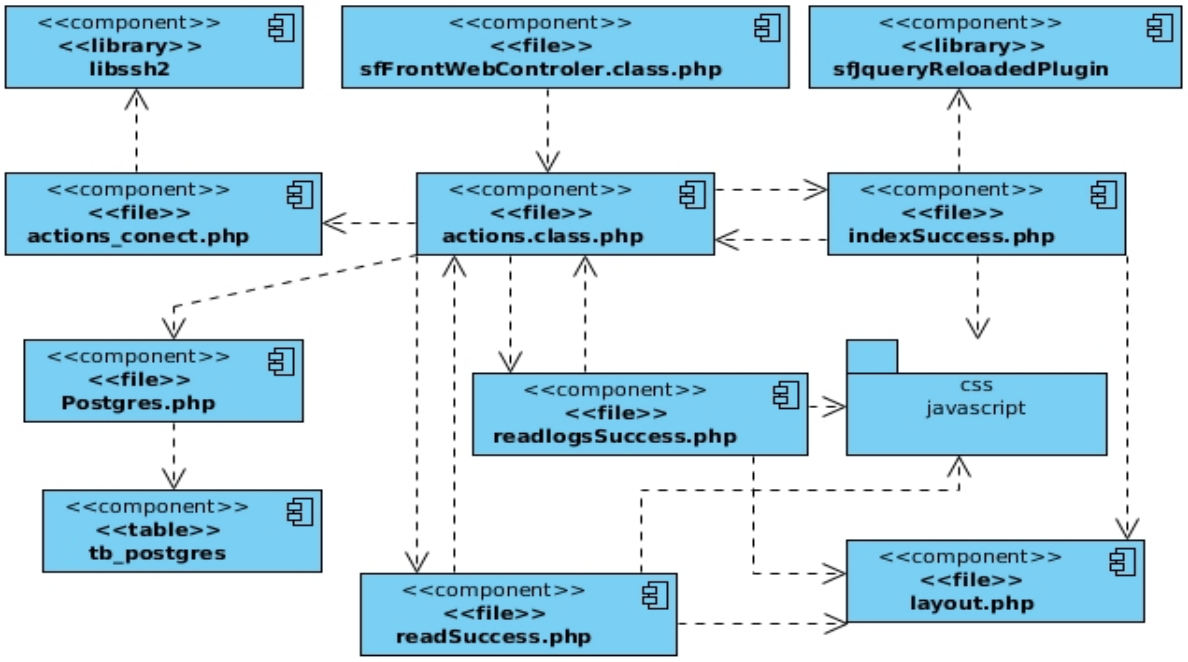


Figura 18: Diagrama de componentes - Módulo PostgreSQL

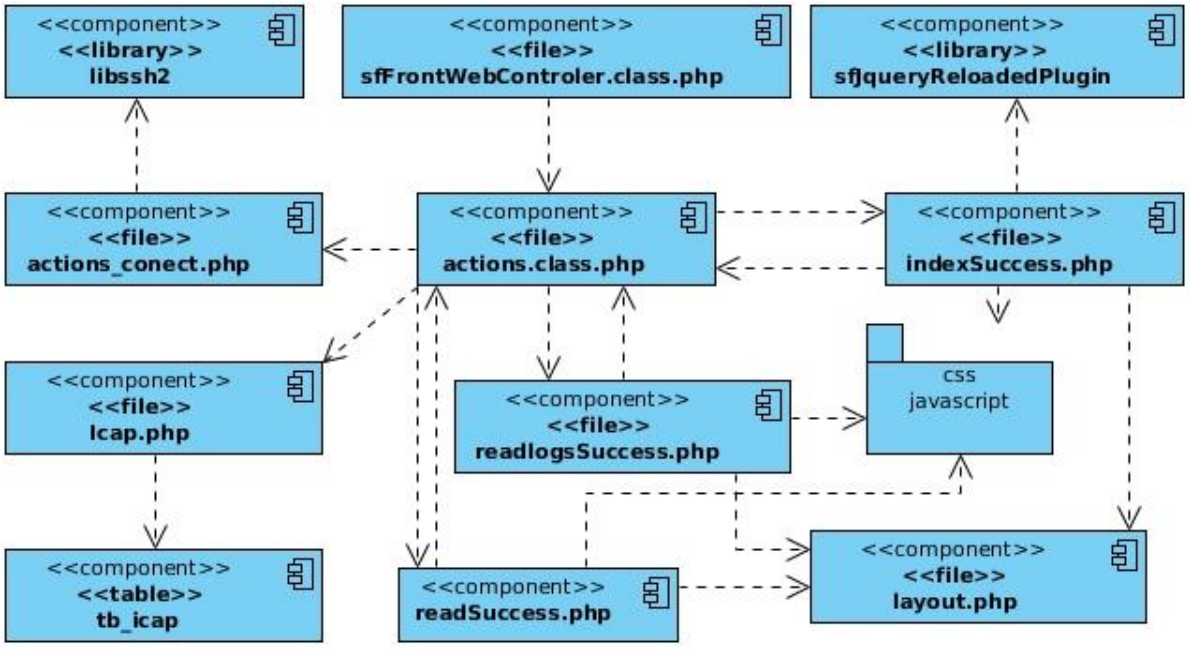


Figura 19: Diagrama de componentes - Módulo Icap

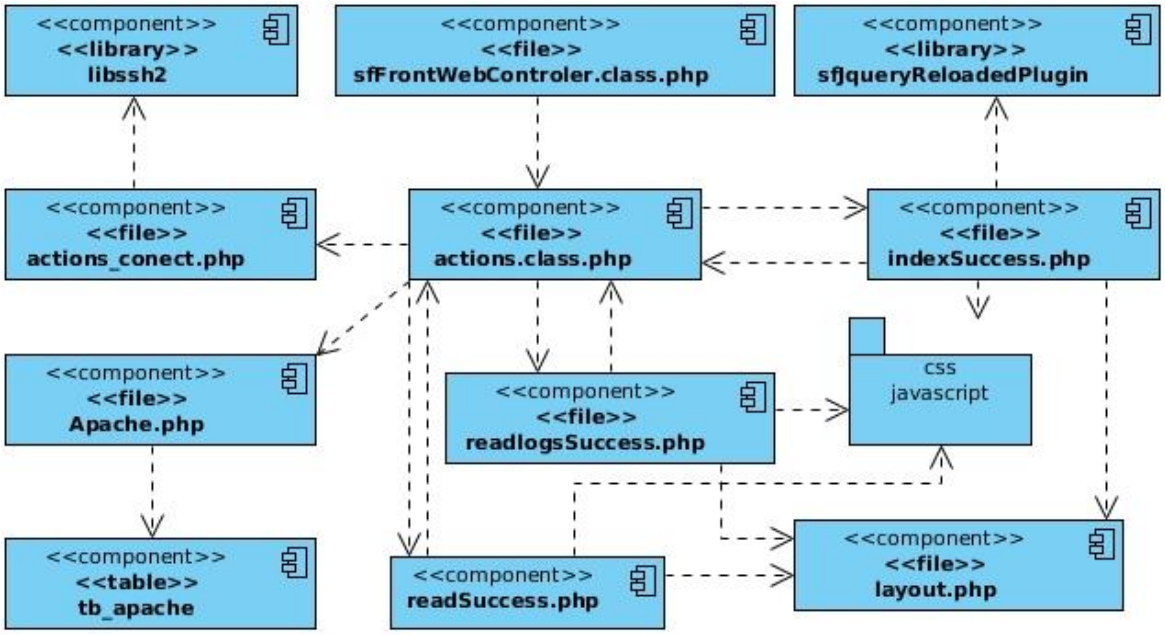


Figura 20: Diagrama de componentes - Módulo Apache

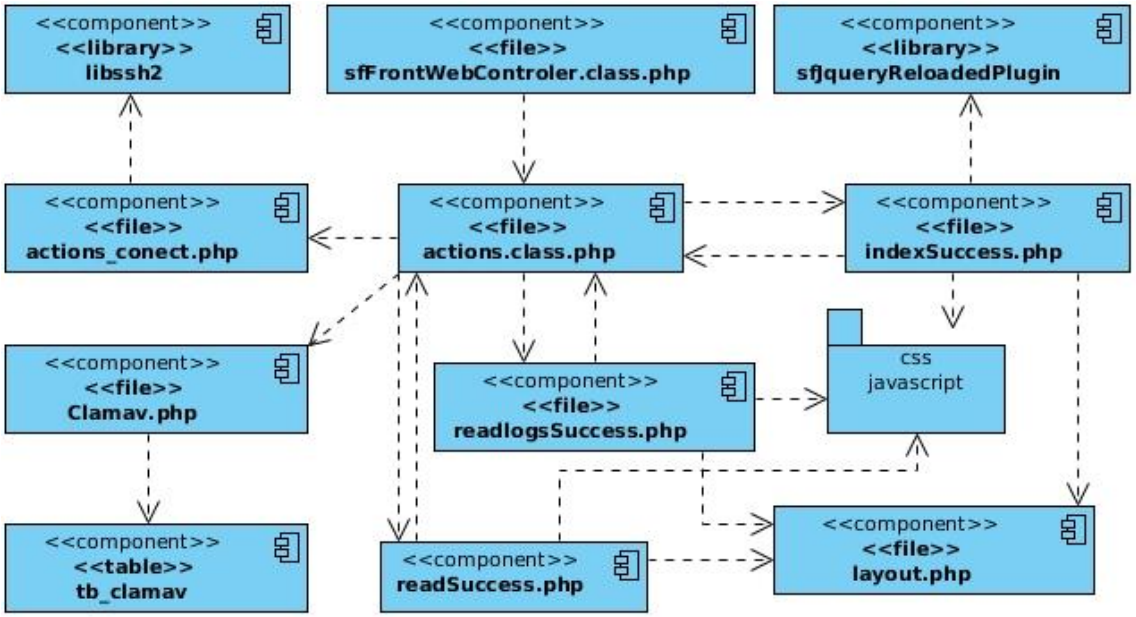


Figura 21: Diagrama de componentes - Módulo Clamav

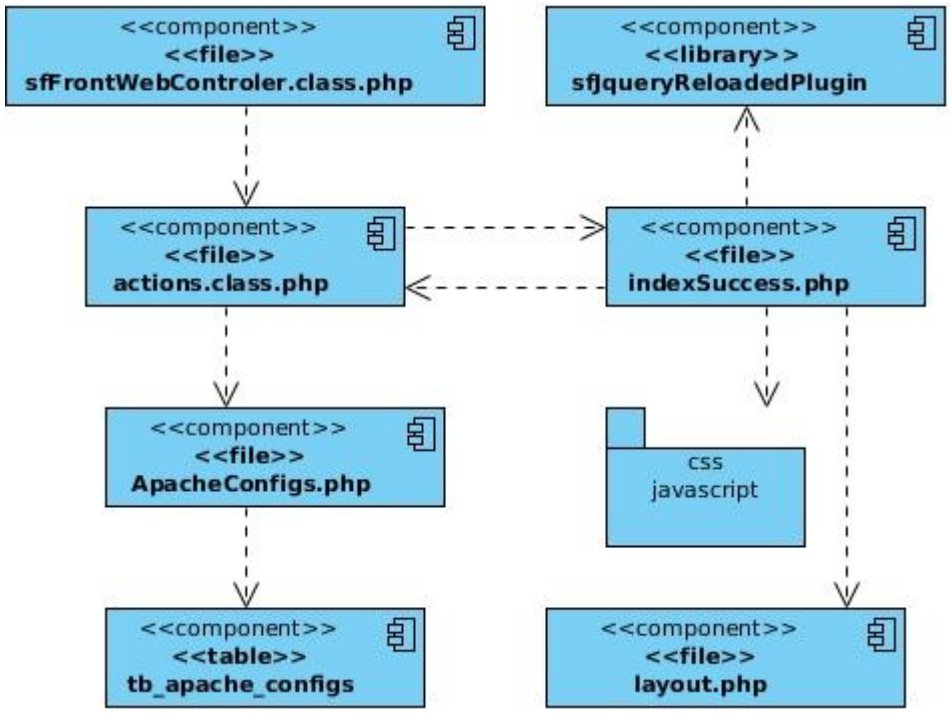


Figura 22: Diagrama de componentes - Módulo apache_configs

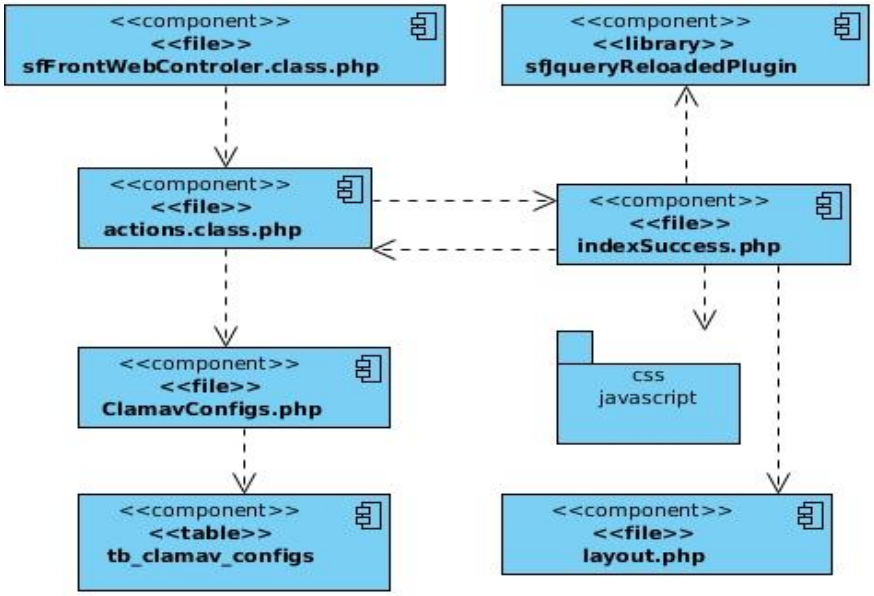


Figura 23: Diagrama de componentes - Módulo clamav_configs

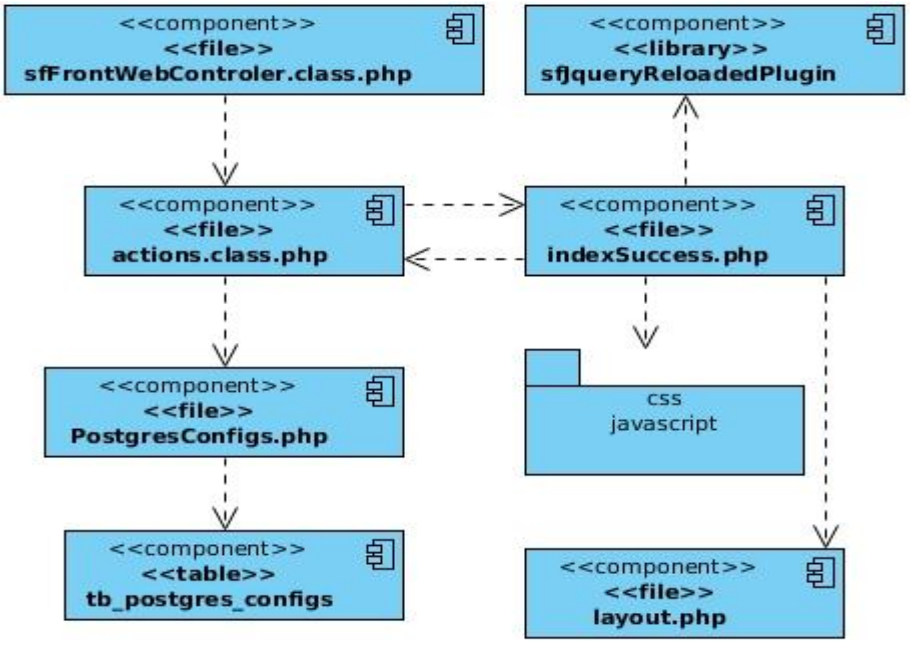


Figura 24: Diagrama de componentes - Módulo postgres_configs

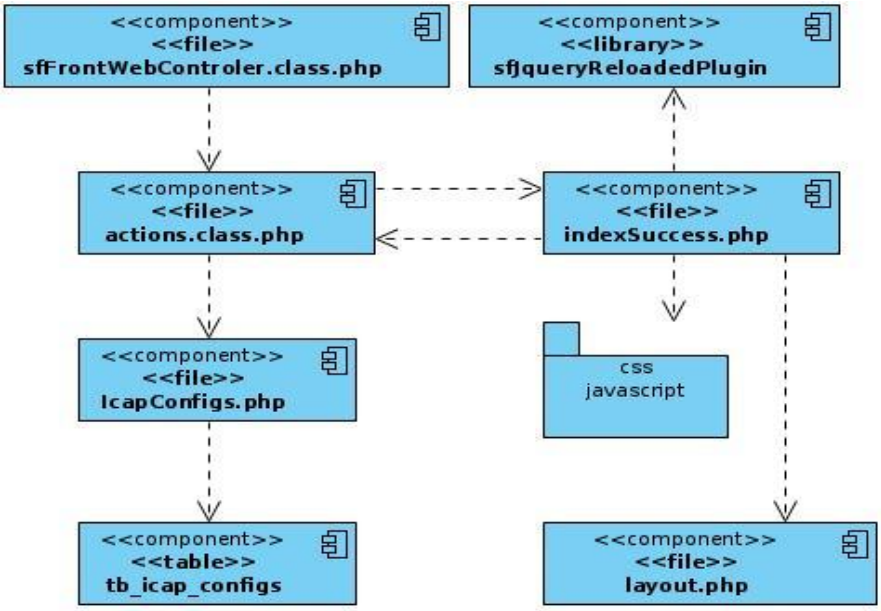


Figura 25: Diagrama de componentes - Módulo icap_configs

Anexo F

Descripción de la clase actions_conect

A continuación se muestra una descripción de los métodos de la clase actions_conect, la cual es utilizada para realizar las conexiones remotas utilizando protocolo SSH.

Método	Descripción	Parámetros de entrada	Retorna
login	Crea una conexión a un servidor remoto a través del protocolo SSH.	**ip**: Dirección del servidor **usuario**: Usuario de acceso al servidor **password**: Contraseña de acceso para el usuario en el servidor.	Verdadero en caso de poder realizar la conexión, falso en el caso contrario.
execute_comand	Ejecuta un comando en el servidor remoto.	**cmd**: Comando a ejecutar.	Retorna una cadena de texto con la salida del comando ejecutado.
execute_comand_array	Ejecuta un comando en el servidor remoto.	**cmd**: Comando a ejecutar.	Retorna un arreglo con la salida del comando ejecutado.
read_config	Lee un fichero de configuración pasado por parámetros verifica si existe.	**config**: Fichero de configuración a leer.	Retorna el fichero de configuración volcado en un arreglo.

send_config_sudo	Guarda un fichero de configuración utilizando sudo en un servidor remoto.	**file**: Dirección del fichero de configuración a salvar en el servidor. **data**: Arreglo con toda la información a adicionar al fichero de configuración. **password**: Contraseña de acceso para utilizar sudo.	
send_config	Guarda un el fichero de configuración sin necesidad de utilizar sudo.	**file**: Dirección del fichero de configuración a salvar en el servidor. **data**: Arreglo con toda la información a adicionar al fichero de configuración.	
get_file	Copia un fichero desde un servidor remoto al sistema de ficheros local.	**local_file:** Fichero local. **remote_file:** Fichero remoto.	Retorna Verdadero en caso satisfactorio, falso en caso contrario.
send_file	Copia un fichero desde el sistema de ficheros local a un servidor remoto.	**local_file:** Fichero local. **remote_file:** Fichero remoto.	Retorna Verdadero en caso satisfactorio, falso en caso contrario.
make_dir	Crea un directorio en un servidor remoto.	**dirname:** Dirección del nuevo directorio en el servidor remoto.	Retorna Verdadero en caso satisfactorio, falso en caso contrario.

rename_file	Renombra un fichero en un servidor remoto.	**file:** Fichero a renombrar. **new_file:** Nuevo nombre para el fichero.	Retorna Verdadero en caso satisfactorio, falso en caso contrario.
remove_dir	Borra un directorio en un servidor remoto.	**dirname:** Dirección del directorio remoto a borrar.	Retorna Verdadero en caso satisfactorio, falso en caso contrario.
remove_file	Borra un fichero en un servidor remoto.	**filename:** Dirección del fichero remoto a borrar.	Retorna Verdadero en caso satisfactorio, falso en caso contrario.
make_symlink	Crea un enlace simbólico en un servidor remoto.	**target:** Dirección del fichero **link:** Dirección del enlace simbólico	Retorna Verdadero en caso satisfactorio, falso en caso contrario.

Glosario de términos

- **3DES**: Triple DES (3DES) se basa en tres iteraciones sucesivas del algoritmo DES, con lo que se consigue una longitud de clave de 128 bits y que es compatible con DES simple.
- **Blowfish:** En criptografía, Blowfish es un codificador de bloques simétricos, diseñado por Bruce Schneier en 1993 e incluido en un gran número de conjuntos de codificadores y productos de cifrado. Es un algoritmo de uso general, que intentaba reemplazar al antiguo DES.
- **Cifrado SSL**: *Secure Sockets Layer*. Protocolo diseñado por la empresa Netscape para proveer comunicaciones encriptadas en Internet.
- **Cliente-Servidor:** La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.
- **Debian:** Es un sistema operativo libre, utiliza el núcleo Linux (el corazón del sistema operativo), pero la mayor parte de las herramientas básicas vienen del Proyecto GNU; cuenta con 29000 paquetes, programas precompilados distribuidos en un formato que hace más fácil la instalación.
- **DES** (*Data Encryption Standard*): Es un estándar de cifrado, es decir, un algoritmo o método para encriptar datos o información, desarrollado originalmente por IBM a solicitud del Oficina Nacional de Estandarización y posteriormente modificado y adoptado por el gobierno de ese mismo país en 1977 como estándar de cifrado de todas las informaciones sensibles no clasificadas.
- **Diffie-Hellman:** Este algoritmo de cifrado de Whitfield Diffie y Martin Hellman supuso una verdadera revolución en el campo de la criptografía, ya que fue el punto de partida para los sistemas asimétricos, basados en dos claves diferentes, la pública y la privada. Vio la luz en 1976.
- **DSA** (Digital Signature Algorithm): Es una parte el estándar de firma digital DSS (*Digital Signature Standard*). Este algoritmo, data de 1991, es una variante del método asimétrico de ElGamal.
- **Escritorio X Window**: El sistema X Window es el estándar gráfico para Linux como visualización de escritorio. Fue desarrollado en el MIT en 1984 y prácticamente todos los sistemas UNIX tienen una versión del mismo. Linux trae una versión denominada XFree86. Normalmente, el X Window

es una capa gráfica intermedia que confía a otra capa denominada gestor de ventanas la visualización de sus elementos.

- **Exploit:** Es una pieza de *software*, o una secuencia de comandos con el fin de causar un error o un fallo en alguna aplicación, a fin de causar un comportamiento no deseado o imprevisto en los programas informáticos, *hardware*, o componente electrónico (por lo general computarizado).
- **Gnome:** Es un acrónimo de 'GNU *Network Object Model Environment*', entorno de trabajo en red orientado a objetos, por lo que Gnome forma parte del más amplio proyecto GNU. Gnome es un entorno de escritorio amigable que permite a los usuarios usar y configurar sus ordenadores de una forma sencilla.
- **GNU/Linux:** Es un poderoso y sumamente versátil sistema operativo con licencia libre y que implementa el estándar POSIX (acrónimo de *Portable Operating System Interface*, que se traduce como Interfaz de Sistema Operativo Portable). En 1992, el núcleo Linux fue combinado con el sistema GNU. El Sistema Operativo formado por esta combinación se conoce como GNU/Linux.
- **GNU:** Es un acrónimo recursivo que significa “GNU no es Unix” (*GNU is Not Unix*). Este proyecto fue iniciado por Richard Stallman, y anunciado el 27 de septiembre de 1983, con el objetivo de crear un sistema operativo completamente libre.
- **IDEA:** Sistema criptográfico simétrico, creado en 1990 por Lai y Massey, que trabaja con bloques de texto de 64 bits, operando siempre con números de 16 bits usando operaciones como OR-Exclusiva y suma y multiplicación de enteros.
- **Módulo**: Es un complemento de *hardware* o *software* que añade una característica o un servicio específico a un sistema más grande.
- **PEAR** (*PHP Extensión and Application Repository*): es un entorno de desarrollo y sistema de distribución para componentes de código PHP. El proyecto PEAR fue fundado por Stig S. Bakken en 1999 para promover la reutilización de código que realizan tareas comunes.
- **Ping:** El comando *ping* permite enviar paquetes ICMP (*Internet Control Message Protocol*) del tipo ECHO_REQUEST a otra computadora, con el objetivo de saber si esta es alcanzable a través de la red. Además, muestra un resumen estadístico acerca del porciento de paquetes perdidos y las velocidades de trasmisión.

- **Rhost:** Es un fichero que contiene una lista de combinaciones host-usuario de confianza para un sistema remoto. Si una combinación host-usuario se encuentra en este archivo, se le otorga al usuario especificado permiso para iniciar sesión de manera remota desde el host sin tener que proporcionar una contraseña.
- **RSA:** El algoritmo de llave pública RSA fue creado en 1978 por Ron Rivest, Adi Shamir y Len Adleman, del Instituto Tecnológico de Massachusetts (MIT); las letras RSA son las iniciales de sus apellidos y es el sistema criptográfico asimétrico más conocido y usado.
- **Setuid:** En Unix existen tres bits de permisos especiales que pueden ser asignados a directorios y/o ficheros ejecutables, *setuid* (*set user information*), setgid (*set group information*) y *sticky*. EL bit *setuid* es asignable a ficheros ejecutables, y permite que cuando un usuario ejecute dicho fichero, el proceso adquiera los permisos del propietario del fichero ejecutado.
- **Spoofing:** Creación de tramas utilizando una dirección falseada; la idea de este ataque es muy sencilla: desde su equipo, se simula la identidad de otra máquina de la red para conseguir acceso a recursos de un tercer sistema que ha establecido algún tipo de confianza basada en el nombre o la dirección IP del ordenador suplantado.
- **TCP/IP:** Es el protocolo base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN).
- **X.509:** Es un estándar ITU-T (estandarización de Telecomunicaciones de la *International Telecommunication Union*) para infraestructura de claves públicas (PKI, o *Public Key Infrastructure*). Entre otras cosas, establece los estándares para certificados de claves públicas y un algoritmo para validación de ruta de certificación. Este último se encarga de verificar que la ruta de un certificado sea válida bajo una infraestructura de clave pública determinada. Es decir, desde el certificado inicial, pasando por certificados intermedios, hasta el certificado de confianza emitido por una Autoridad Certificadora (CA, o *certification Authority*).