Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Desarrollo de una nueva versión del sistema GAMSW para la configuración del servicio web Apache en el Departamento de Componentes del Centro Telemática.

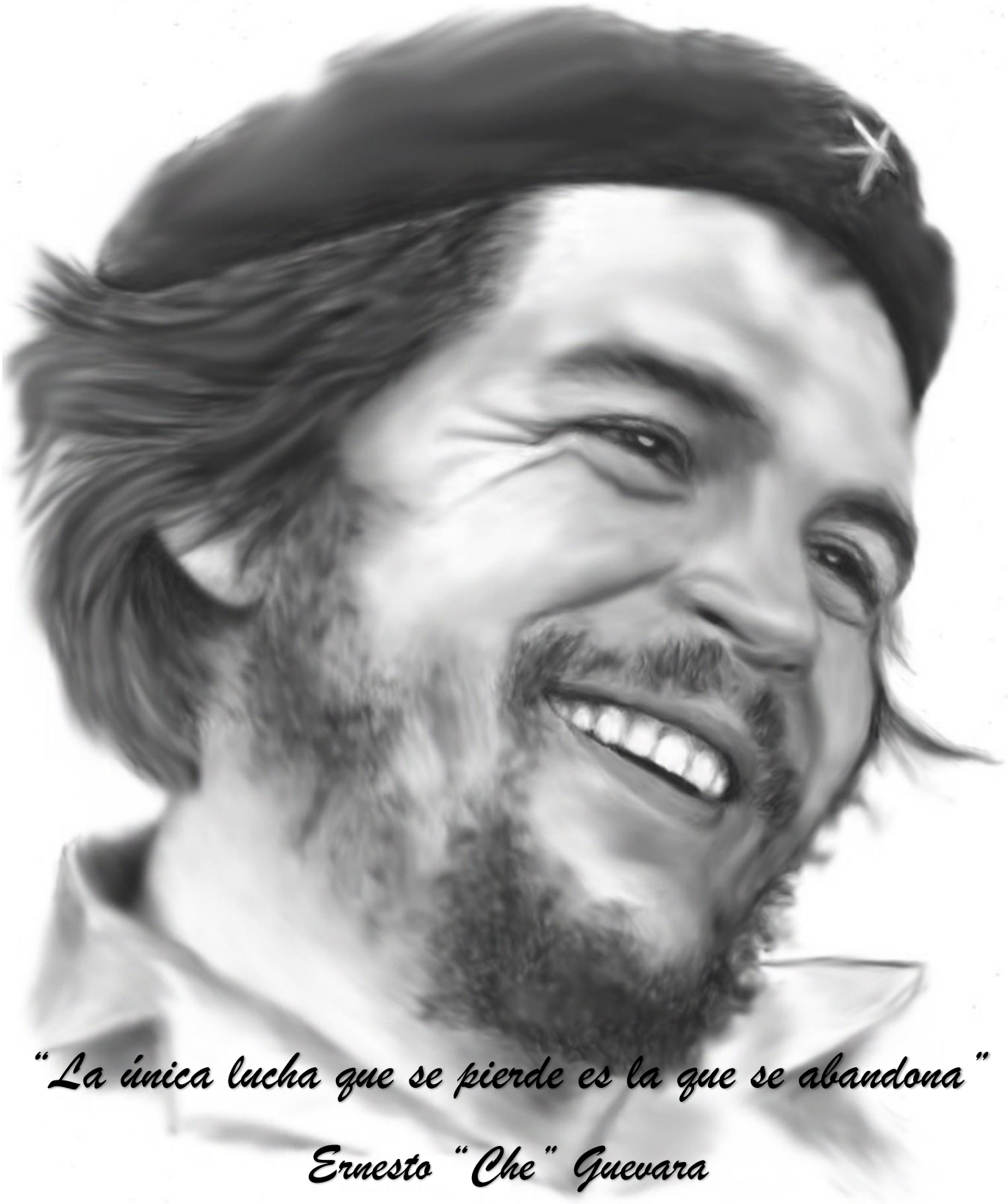
Autor: Mario William Fernández Gómez

Tutores: Drc.Arturo Orellana García

Ing. Osmar Capote Vázquez

Ing. Roberto Antonio Infante Milanés

La Habana, noviembre del 2019



"La única lucha que se pierde es la que se abandona"

Ernesto "Che" Guevara

Declaración de autoría

Declaro ser autor de la presente tesis que tiene por título: “Desarrollo de una nueva versión del sistema GAMSW para la configuración del servicio web Apache en el Departamento de Componentes del Centro Telemática” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Mario William Fernández Gómez

_____

Firma del autor

Ing. Osmar Capote Vázquez

_____

Firma del tutor

Ing. Roberto Antonio Infante Milanés

_____

Firma del tutor

Drc.Arturo Orellana García

_____

Firma del tutor

Datos de contacto

Autor: Mario William Fernández Gómez

Universidad de las Ciencias Informáticas

Email: mwfernandez@estudiantes.uci.cu

Tutor: Ing. Roberto Antonio Infante Milanés

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de Granma, en el año 2015. Profesor del Departamento de Ingeniería y Gestión de Software de la Facultad, desde septiembre del 2015. Ha impartido las asignaturas de Sistemas de Bases de Datos 1 y 2 y de Programación 1.

Email: rainfantem@uci.cu

Tutor: Ing. Osmar Capote Vázquez

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas, en el año 2015. Desarrollador del Centro de Telemática. Actualmente es el jefe del Departamento de Desarrollo de Componentes de dicho centro.

Email: ocapote@uci.cu

Tutor: Dr.C. Arturo Orellana García:

Dedicatoria

Quisiera dedicarle esta tesis, en primer lugar, a mi abuelo William, por ser mi guía, mi ejemplo a seguir, mi amigo y maestro de la vida. Por ser la infinita fuente de conocimientos que siempre guio mis pasos por los caminos del saber.

A mi madre por ser el zapador de mis andanzas, por ir siempre delante por los caminos de la vida removiendo piedras y obstáculos para que yo no tuviera que tropezar, sufriendo en muchos casos ella los daños por mí.

A mi novia Beatriz por haber sido todos estos años la causa de mis risas después de los llantos, por haber sido mi oasis en los desiertos, mi pilar de apoyo en los momentos más difíciles y mi felicidad en los momentos más tristes.

A mi papá Francisco por su papel como pionero en mis primeros pasos, por haber sido mi figura representativa ante cada problema y dificultad que pude tener desde pequeño.

A mis hermanas por ser las flores de mi jardín, esas flores sobre las que nos esforzamos en su cuidado a diario tan solo por levantarnos en la mañana verlas despertar por los rayos del Sol.

A toda mi familia por ser la amalgama de experiencias que me ha evitado tantos tropiezos y desilusiones a lo largo de todos estos años, por ser mi más grande recurso educativo y la razón de mi ser.

A mis amigos por ser el motor impulsor de mis buenos actos y por aceptar ser mi familia escogida.

Agradecimientos

Quisiera comenzar agradeciéndole a mi abuelo Williams por todas las enseñanzas y valores que me fue inculcado desde pequeño. Por haberme enseñado gran parte de lo que sé, aunque fuera usando esa frase que tanto odiaba cuando en mi afán por no hacer algo le decía "no puedo hacer eso porque no se", me miraba fijamente y entre risas decía "es tu día de suerte, hoy vas a aprender". Gracias abuelo por siempre estar ahí para mis dudas, para los consejos, para las risas y los regaños.

A mi mami gracias por todo lo que has hecho, lo que haces y lo que sé que seguirás haciendo para que tus niños sigan adelante sin tener tropiezos. La vida te ha puesto muchos obstáculos y soy testigo de todo lo que has pasado para que hoy yo pueda estar aquí. Gracias por todo el sacrificio, por esas mañanas en las que te levantabas luchando contra el sueño y el cansancio para prepararme la merienda de la escuela. Gracias mami por enseñarme a nunca rendirme ante ninguna adversidad, porque tú nunca lo hiciste. Eres la persona más fuerte que conozco.

A mi hermana Sandra por todo el apoyo incondicional en todos estos cursos, por su amor infinito y sus frases de aliento que me ayudaron a seguir adelante cuando yo mismo no creí en mí. Gracias tata por ser como eres, sabes que sin ti nada de esto hoy fuera posible.

A mi cuñado Alejandro por no dudar ni un segundo desde el primer momento que nos conocimos e incluirme como parte de su familia. Gracias mi hermano por cada consejo, cada abrazo y por todo tu apoyo.

Al amor de mi vida, a mi compañera de sentimientos, a mi razón de existir. Gracias mi amor por estar siempre ahí en los momentos que más te necesité, por quererme por quién soy, por permanecer a mi lado cuando no hemos tenido nada. Gracias por cada beso, por cada perdón, por cada lección que me diste sin decir una palabra. Gracias por cada momento vivido, por cada abrazo en mis momentos de furia y tristeza. Siempre supiste que no iba a ser fácil, que me ibas a extrañar e incluso que no siempre que me necesitaras iba a poder estar y aun así permaneciste siempre junto a mí. Gracias tata por todo el amor que me has dado.

A mi papá Francisco, mi primer guía, gracias por quererme siempre como tu niñito, no sabría qué sería de mi si un día no llega a ser así. Gracias por todo lo que me enseñaste, por tolo que por mi hiciste, pero por sobre todo gracias por quererme.

A mis abuelas Tati, Cary, Ilda y Julia gracias por todos los mimos, por todos los consejos, los cafecitos, las sopas cuando he estado enfermo. Gracias por cada atención, por cada llamada, por taparme con la corcha desde chico en las noches de frío. Las quiero mucho.

A mi tío Francisco por todo el cariño brindado, por cada travesía y por cada abrazo sincero, esos de los que te hacen sentir que pase lo que pase nunca estarás solo. La distancia nos ha mantenido separados, pero nunca seremos desunidos, eres como un padre para mí.

A Pedro, quién llegó de ultimo a la manada. La vida nos pone tropiezos, no todo es fácil, pero nada es imposible. Tienes a tu lado un gran ejemplo a seguir. Gracias por hacer de esta tu familia, sé que te esfuerzas mucho y por eso te has ganado mi cariño, amor y respeto. Te quiero mucho papá.

A mis amigos gracias por cada momento compartido, por cada abrazo sincero, por cada consejo, por cada charla, por cada chucho, son ustedes los que permiten que cada problema, cada revés y cada fracaso sea lo menos importante en el momento que estamos juntos.

A mi hermanita menor y mis primos, las personitas que me hacen sentir grande y sabio con cada duda y consejo que me piden. Gracias por ver en mí un ejemplo a seguir. Los quiero mucho.

Resumen

La administración y mantenimiento de equipos y servicios dedicados a guardar y brindar datos es un factor de gran importancia para todo negocio cuyo crecimiento depende del manejo eficiente de la información. Es de gran importancia en estos casos lograr la realización rápida de procesos como el acceso y la gestión de la información. En la actualidad la UCI cuenta con el Departamento de Componentes del centro Telemática en el cuál se hace uso frecuente de la configuración de servicios web, entre los cuales se encuentra el servicio web Apache. Para la realización de dicho proceso el departamento cuenta con el sistema Gestión Administración y Monitoreo de Servicios Web (GAMSW), el mismo permite la gestión, administración y monitoreo de servidores web apache de manera remota en computadoras servidoras conectadas a la red. A pesar de estas funciones, en la actualidad el sistema carece de funcionalidades que faciliten el trabajo de los especialistas y permita realizar algunos procesos de manera fácil, situación que provoca la realización lenta y engorrosa de los mismos y que posibilita la ocurrencia de errores humanos durante su ejecución. En la presente investigación se desarrolla la versión 2.0 del sistema GAMSW, agregando funcionalidades que permitan mostrar la información de los log de manera gráfica, modificar parámetros del archivo principal de Apache que se muestran en un formulario, brindar una consola de administración a los especialistas y realizar un balance de carga entre servidores web Apache, dando solución así a parte de los problemas existentes en el departamento.

Palabras Claves: administración, mantenimiento, servicios, datos, información, sistema, monitoreo, servidores web apache, computadoras servidoras, GAMSW, funcionalidades.

Abstract

The administration and maintenance of equipment and services dedicated to saving and providing data is a very important factor for any business whose growth depends on the efficient management of information. It is of great importance in these cases to achieve the rapid realization of processes such as access and information management. At present, the UCI has the Components Department of the Telematic Center in which frequent use is made of the configuration of web services, among which is the Apache web service. To carry out this process, the department has the Management System and Monitoring of Web Services (GAMSW), which allows management, administration and monitoring of Apache web servers remotely in server computers connected to the network. In spite of these functions, at present the system lacks functionalities that facilitate the work of the specialists and allows to carry out some processes in an easy manner, situation that causes the slow and cumbersome realization of them and that allows the occurrence of human errors during its occurrence. In the present investigation the 2.0 version of the GAMSW system is developed, adding functionalities that allow to show the information of the log in a graphic way, modify parameters of the Apache main file that are shown in a form, provide an administration console to the specialists and perform a load balance between Apache web servers, thus providing a solution to part of the existing problems in the department.

Keywords: administration, maintenance, services, data, information, system, monitoring, apache web servers, server computers, GAMSW, functionalities.

Índice

Índice de tablas

Índice de figuras

Introducción

El ser humano cada día aprovecha más las potencialidades que le proporciona las Tecnologías de la Información y las Comunicaciones (TIC por sus siglas en español), debido a que le permite lograr una agilización de procesos complejos llegando a obtener un alto nivel de ahorro en tiempo y recursos. Un ejemplo de estas tecnologías son los servidores, componente responsable del avance y la transformación que ha sufrido la comunicación con el paso de los años.

Estos servidores son uno de los factores clave del almacenamiento y la transformación digital. Es el equipo informático encargado de transmitir la información para el correcto funcionamiento de la organización y tiene funciones base para los servicios que brinda ya que es donde se ejecutan estos. Muchos de estos son conocidos como servidores web, componente que permite que toda la información publicada en cada sitio web se almacene en un espacio destinado para este fin, permitiendo así la divulgación de su contenido. La arquitectura utilizada por este tipo de servidores es la conocida como cliente/servidor, es decir, el equipo cliente hace una solicitud o petición al equipo servidor y este atiende dicha solicitud (Baxter et al. 2008). Para lograr sus propósitos, los servidores cuentan con varios servicios dentro de los cuales se encuentran los servicios web, los que cumplen con el rol fundamental de estandarizar la comunicación entre distintas plataformas y lenguajes de programación.

Uno de los servicios web más antiguos y confiables es el Apache, con la primera versión lanzada en 1995 y actualmente se desarrolla dentro del proyecto HTTP *Server de Apache Software Fundation* (Kabir 2004). Su trabajo es establecer una conexión entre un servidor y los navegadores de los visitantes del sitio web (*Firefox, Google Chrome, Safari*, entre otros.) mientras envían archivos entre ellos (estructura cliente-servidor). Apache es un *software* multiplataforma, por lo cual funciona tanto en servidores Unix como en Windows.

El servicio web Apache es altamente personalizable y por las ventajas que brinda es objeto de uso en Cuba. El dominio de las TIC en Cuba ha sido muy necesario para poder introducirlas en la práctica social y lograr un desarrollo sostenible. Para esto, el país se ha propuesto el despliegue de dichas tecnologías como uno de los pilares del programa de informatización de la sociedad cubana. Para el despliegue y uso de estas tecnologías, el país se ha apoyado en diversas instituciones como es el caso de la Universidad de Ciencias Informáticas (UCI). Dicha institución tiene dentro de sus objetivos fundamentales el de desarrollar la industria del *software* para contribuir al desarrollo social y económico del país (Márquez y Rosa 2016) .La universidad

cuenta con varios centros de desarrollo, entre ellos se encuentra el Centro de Telemática (TLM). En este centro se desarrollan sistemas y servicios informáticos integrales para las Telecomunicaciones y la Seguridad Informática (Hernández Dominguez 2018).

En la actualidad, en el Departamento de Componentes del Centro Telemática se cuenta con una aplicación web de nombre GAMSW que tiene como objetivo integrar las funcionalidades del ecosistema del servidor web Apache, pero la misma no cuenta actualmente con algunas características funcionales de dicho ecosistema que facilitarían el trabajo a los administradores del departamento. En este, los administradores aceden a los *log* (registros) del servidor y ven la información de los mismos a través de la consola, lo que trae consigo una pérdida considerable de tiempo debido a que el especialista tiene que hacer uso de códigos de forma manual. La ejecución de comandos por consola se torna engorrosa a la hora de trabajar con los servidores y aun así existen administradores que prefieren utilizarla debido a las habilidades que presentan mediante el uso de la misma, pero mientras trabajan con la aplicación GAMSW tienen que utilizar una consola ajena al sistema, lo cual propicia que trabaje con varias aplicaciones simultáneamente e ir usando códigos en varios lugares por separado. En múltiples ocasiones algunos de los servidores del centro se encuentran saturados de peticiones que influyen de manera negativa en su óptimo funcionamiento, mientras otros se encuentran brindando una cantidad mínima de servicios aun teniendo la capacidad de compartirse el trabajo entre ellos. El servicio Apache cuenta con un archivo principal conocido como apache2.conf y en el Departamento de Componentes del Centro Telemática se configuran con frecuencia diversos parámetros de este archivo, pero no se realiza de manera rápida y sencilla ya que no se puede hacer de forma gráfica sino a través de un terminal introduciendo comandos. Esto trae consigo que el proceso sea complejo tecnológicamente para los especialistas y que pueda conllevar a la ocurrencia de errores humanos durante su ejecución.

Por lo anteriormente planteado, se identifica el siguiente **problema a resolver:** ¿Cómo gestionar de forma centralizada las configuraciones de los servicios web Apache del Departamento de Componentes del centro TLM?

El **objeto de estudio** del presente trabajo es: el proceso de configuración de servicios web Apache para Ubuntu y Debian.

Para la solución de este problema se define como **objetivo general**: Desarrollar una nueva versión de la aplicación web GAMSW del Departamento de Componentes del Centro Telemática, que permita gestionar de forma centralizada las configuraciones de los servicios web Apache para Ubuntu y Debian.

El **Campo de acción** se encuentra delimitado a**:** las configuraciones de los servicios web Apache para Ubuntu y Debian del Departamento de Componentes del Centro Telemática.

En la presente investigación se propone las siguientes **preguntas científicas:**

1. ¿Qué referentes teóricos-metodológicos sustentan al proceso de configuración de servicios web Apache en el mundo?
2. ¿Qué características tienen los sistemas de configuración de servicios web que constribuyan a dicho proceso en el departamento de Componentes del Centro Telemática?
3. ¿Cómo organizar el proceso de evolución del sistema GAMSW que contribuya a la realización del proceso de configuración del servicio web Apache en el Departamento de Componentes den centro telemática?
4. ¿Cómo validar que el sistema evolucionado cumple con las necesidades detectadas en la problemática antes planteada?

Para dar respuesta a las preguntas científicas se desglosan las siguientes **tareas de investigación**:

1. Elaboración del marco teórico conceptual de la investigación relacionado al proceso de configuración de servicios web Apache para Ubuntu y Debian y los elementos fundamentales del campo de acción.
2. Análisis y diseño de las funcionalidades asociadas al proceso de configuración de servicios web Apache para Ubuntu y Debian aplicando la metodología AUP-UCI.
3. Desarrollo de una nueva versión del sistema GAMSW, teniendo en cuenta las características de los servicios web Apache del Departamento de Componentes del centro TLM.
4. Realización de pruebas para validar el código y el correcto funcionamiento de la aplicación, teniendo en cuenta la satisfacción del cliente.

En el transcurso de la investigación se utilizaron los siguientes **métodos científicos**:

Para guiar el desarrollo de la investigación se utilizaron métodos teóricos y empíricos de acuerdo a los autores (Hernández León y Coello González 2012).

Métodos teóricos

Permiten estudiar las características del objeto de investigación que no son observables directamente. Facilitan la construcción de modelos. Crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad. Se apoyan en el proceso de análisis y síntesis (Hernández León y Coello González 2012).

Métodos empíricos

Describen y explican las características fenomenológicas del objeto. Representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional (Hernández León y Coello González 2012).

En el transcurso de la investigación se utilizaron los siguientes **métodos teóricos**:

1. Analítico-Sintético: se ve reflejado en la investiga a partir del análisis en estado del arte, en la justificación de las tecnologías y herramientas empleadas, en los diferentes elementos del campo de acción.

2. Inductivo-Deductivo: para la identificación de la problemática y de las soluciones, se evidencia en las conclusiones parciales y generales de la investigación, así como en los resultados derivados del análisis en las distintas etapas de la investigación.

3. Modelación: se utilizó para obtener los artefactos ingenieriles asociados a la metodología AUP-UCI según el escenario número 4. Permitió comprender el problema y su solución a partir de la elaboración de esquemas y diagramas.

Además, se empleó la entrevista como **método empírico**. Fue realizada a los especialistas del Centro de Telemática para comprender el contexto del problema y definir los elementos necesarios a tener en cuenta en la propuesta de solución.

El presente documento consta de tres capítulos, las conclusiones generales, la bibliografía general utilizada y un glosario de términos. La estructura de los capítulos se define de la siguiente forma:

Capítulo 1. Fundamentos teóricos y metodológicos de la configuración de servicios web Apache

En este capítulo se establecen las bases teóricas de la investigación, se exponen los principales conceptos asociados a la configuración de servicios web Apache y se realiza el análisis del estado del arte de sistemas homólogos para establecer los elementos técnicos y funcionales en la propuesta de solución.

Capítulo 2. Sistema de configuración de servicios web Apache

Se realiza una caracterización del componente a desarrollar, así como la descripción de las funcionalidades a implementar. Se define el modelo de dominio, los requisitos funcionales y no funcionales, así como los actores, clases y patrones de diseño. Además, se realiza la descripción de los requisitos funcionales propuestos.

Capítulo 3. Implementación y validación de la propuesta de solución

Se especifican los principales aspectos relacionados con la implementación de la propuesta de solución. Se presentan el modelo de despliegue y el diagrama de componentes, así como el estándar de codificación empleado. Se define la estrategia de pruebas para la validación del sistema GAMSW 2.0 y se documentan los resultados de la misma.

Capítulo 1. Fundamentos teóricos y metodológicos de la configuración de servicios web Apache

Introducción

En este capítulo se definen los elementos teóricos que sustentan la investigación. Se hace referencia a los conceptos asociados a la configuración de servicios web Apache. Se realiza un análisis del estado del arte de sistemas informáticos para la configuración de servicios web Apache a nivel nacional e internacional. Además, se selecciona el ambiente de desarrollo y la metodología a utilizar para el desarrollo del *software*.

1.3. Conceptos asociados a la investigación

En este epígrafe se especifican los principales conceptos asociados al dominio de la investigación, los cuales permiten fundamentar conceptualmente el capítulo uno de este trabajo.

Servicio

Un servicio es el conjunto de actividades que se relacionan entre sí y de actitudes que se diseñan para satisfacer las necesidades de los usuarios. Un servicio es también el resultado de un proceso, es decir, el resultado de llevar a cabo necesariamente al menos una actividad en la interfaz entre el proveedor y el cliente y generalmente es intangible. Es cualquier beneficio que una parte ofrece a otra y su producción puede estar vinculada o no con un producto físico (Pérez Ríos 2014).

Servicio Web

El servicio *World Wide Web* o simplemente web, se podría definir como un amplio sistema multimedia de acceso a información distribuida por toda la red en forma de documentos hipertextuales. La información reside en forma de páginas web en ordenadores que se denominan servidores web. Es una aplicación *software* identificada por un URI (*Uniform Resource Identifier*) cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML (Lenguaje-Marcado-Extensible) intercambiados mediante protocolos de *internet* (Monsalve 2015).

Servicio web Apache

El servicio web Apache HTTP es un servicio de código abierto para una multitud de plataformas y de sistemas operativos. Permite la creación de páginas y servicios web. Es altamente configurable, admite 9 bases de datos de autenticación y negociado de contenido, aunque carece de una interfaz gráfica que ayude

en su configuración. Adaptado a los nuevos protocolos, y cuya implementación se realiza de forma colaborativa (JM Pantoja Blyde 2013).

Configuración de servicios web

Es la configuración y mantenimiento de equipos y servicios dedicados a guardar y brindar datos. Su función es muy importante para todo negocio cuyo crecimiento depende del manejo eficiente de la información. La misma implica mantener un servidor informático del tipo que sea actualizado y funcionando correctamente para que una red de ordenadores pueda operar sin ningún tipo de problemas.

Balanceador de carga

El balanceo de carga es la manera en que las peticiones de *Internet* son distribuidas sobre una fila de servidores. Existen varios métodos para realizar el balanceo de carga. Desde el simple "*Round Robin*" (repartiendo todas las peticiones que llegan de *Internet* entre el número de servidores disponibles para dicho servicio) hasta los equipos que reciben las peticiones, recogen información, en tiempo real, de la capacidad operativa de los equipos y la utilizan para enrutar dichas peticiones individualmente al servidor que se encuentre en mejor disposición de prestar el servicio adecuado (Baxter et al. 2008).

Log

Es una grabación secuencial en un registro, ya sea en un archivo o en una base de datos, de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular (aplicación, actividad de una red informática, entre otros), constituyendo así una evidencia del comportamiento del sistema.

1.4. Proceso de configuración de servicios web Apache para Ubuntu y Debian.

En Ubuntu y Debian, Apache mantiene sus archivos de configuración principales dentro de la carpeta "/ *etc* / *apache2*". En esta se encuentran los archivos *apache2.conf, envvars, magic, mods-enabled/, sites-available/, conf.d/, httpd.conf, mods-available/, ports.con y sites-enabled*. *Apache2.conf* es el archivo de configuración principal del servidor, desde el cual se pueden realizar casi todas las configuraciones. Este archivo configurará los valores predeterminados y será el punto central de acceso para que el servidor pueda leer los detalles de la configuración. Debido a esto, la configuración de Apache no solo tiene lugar en un único archivo monolítico, sino que pasa a través de un diseño modular en el que se pueden agregar y modificar nuevos archivos según sea necesario (Django 2019).

Los detalles de configuración principales del servidor Apache se mantienen en el archivo *"/etc/apache2/apache2.conf".* Este archivo se divide en tres secciones principales: configuración para el proceso global del servidor Apache, configuración para el servidor predeterminado y configuración de hosts virtuales. En Ubuntu y Debian, la mayoría del archivo es para definiciones globales, y la configuración del servidor predeterminado y hosts virtuales se maneja al final, usando la directiva *"Include"*. La misma le permitirá a Apache leer otros archivos de configuración en el archivo actual en la ubicación en la que aparece la sentencia (Django 2019).

Según (Ubuntu 2016) el resultado es que Apache genera dinámicamente un archivo de configuración global al inicio. En la sección de configuración global se encuentran varias opciones que controlan el funcionamiento de Apache en su conjunto. Entre estas opciones se encuentran los parámetros:

- ***KeepAlive:*** Esta opción, si está establecida en *"On"*, permitirá que cada conexión permanezca abierta para manejar múltiples solicitudes del mismo cliente. Si está en *"Off"*, cada solicitud tendrá que establecer una nueva conexión, lo que puede resultar en gastos generales significativos dependiendo de su configuración y situación de tráfico.

- ***KeepAliveTimeout:*** Esta configuración controla cuánto tiempo debe esperar la siguiente solicitud después de finalizar la última. Si se alcanza el umbral de tiempo de espera, la conexión morirá.

- ***MaxKeepAliveRequests:*** Esta directiva establece el máximo número de peticiones que se pueden realizar en una conexión persistente. Las conexiones persistentes tienen que estar activadas, obviamente. Hay que tener en cuenta el ancho de banda de salida de nuestro servidor, por el cual deberá ser enviada toda la información. Si se establece un valor muy grande respecto al ancho de banda, el tiempo de respuesta se verá incrementado para cada usuario. El valor predeterminado de la directiva *MaxKeepAliveRequests* es de 100, que debería bastar en la mayoría de los casos.

- ***Timeout:*** Define, en segundos, el tiempo que el servidor esperará para recibir y enviar peticiones durante la comunicación, tras los cuales el servidor cierra la conexión. Está configurado por defecto a 300, lo cual es apropiado para la mayoría de las situaciones.

- ***LogLevel:*** Especifica el tipo de mensajes que se guardarán en el fichero de registro de errores. Dependiendo de los valores especificados, se guardarán más o menos. Esta directiva sólo se

puede encontrar fuera de cualquier sección. Valor de más a menos son: *debug, info, notice, warn, error, crit, alert, emerg.*

- ***ServerRoot:*** Esta directiva define el directorio donde se ubica toda la información de configuración y registro que necesita el servidor para su correcto funcionamiento, como por ejemplo *srm.conf, httpd.conf, acces.conf*, etc. La ubicación que se indique aquí debe ser una ruta absoluta, lo que significa que debemos indicar la ubicación del directorio partiendo desde la raíz.
- ***HostnameLookups:*** puede aparecer en *on* o en *off*. Si el servidor permite la directiva *HostnameLookups* (poniéndolo en *on*), el servidor resolverá automáticamente la dirección IP de cada conexión que pida un documento del servidor.

- **ErrorLog:** Especifica la ubicación del fichero que contiene el registro de errores. Por defecto en la carpeta logs. Esta directiva sólo se puede encontrar fuera de cualquier sección.

Según (CodeASite 2018) los archivos de registro o trazas (*logs*) son útiles en la detección de fallos y sus intentos de solución. En las distribuciones Debían y Ubuntu estos archivos se encuentran en el directorio */var/log*. En el mismo se encuentran los archivos:

/message: registro de mensajes generales del sistema
/auth.log: log de autenticación
/kern.log: registro del *kernel*
/cron.log: registro de *crond*
/maillog: registro del servidor de mails
/qmail/: registro de *Qmail*
/apache2/: registro de errores y accesos a Apache
/lighttpd: registro de errores y accesos a *Lighttpd*
/boot.log: registro de inicio del sistema
/mysqld.log: registro de la base de datos *MySQL*
/secure: *log* de autenticación
/utmp or /var/log/wtmp: registro de *login*

En el directorio */var/log/apache2/* se encuentra el *log* de acceso de nombre access.log. Para las distribuciones Debian y Ubuntu el mismo toma el formato conocido como Formato de registro combinado. Este posee una estructura similar a: *127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif*

HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)" (CodeASite 2018). Cada una de las entradas se describen a continuación:

127.0.0.1 (%h): Es la dirección IP del cliente (*host* remoto) que hizo la petición al servidor. Si la directiva *HostnameLookups* tiene valor *On*, el servidor intentará determinar el nombre del host y registrar ese nombre en lugar de la dirección IP. Sin embargo, no se recomienda que use esta configuración porque puede ralentizar significativamente las operaciones del servidor. En su lugar, es mejor usar un programa que realice esta tarea posteriormente sobre el registro, una variante de este es *logresolve*. Las direcciones IP que se registren no son necesariamente las direcciones *host* de los usuarios finales. Si existe un servidor proxy entre el usuario final y el servidor, la dirección que se registra es la del proxy.

- (%l): Un "guion" significa que la información que debería ir en ese lugar no está disponible. En este caso, esa información es la identidad RFC 1413 del cliente, determinada por *identd* en la PC del cliente. Esta información es poco fiable y no debería ser usada nunca a excepción con clientes que estén sometidos a controles muy estrictos en redes internas. Apache *httpd* ni siquiera intenta registrar esa información a menos que la directiva *IdentityCheck* tenga valor *On*.

frank (%u): Este es el identificador de usuario de la persona que solicita el documento determinado por la autentificación *HTTP*. Normalmente ese mismo valor se pasa a los scripts CGI con la variable de entorno *REMOTE_USER*. Si el código de estado de la petición es 401, entonces no debe confiar en la veracidad de ese dato porque el usuario no ha sido aún autentificado. Si el documento no está protegido por contraseña, se mostrará un guion "-" en esta entrada.

[10/Oct/2000:13:55:36 -0700] (%t): La hora en la que el servidor recibió la petición. El formato es: [día/mes/año:hora:minuto:segundo zona_horaria]

"GET /apache_pb.gif HTTP/1.0" (\"%r\"): La línea de la petición del cliente se muestra entre dobles comillas. La línea de petición contiene mucha información de utilidad. Primero, el método usado por el cliente es *GET*. Segundo, el cliente ha hecho una petición al recurso */apache_pb.gif*, y tercero, el cliente usó el protocolo *HTTP/1.0*. También es posible registrar una o más partes de la línea de petición independientemente.

200 (%>s): Es el código de estado que el servidor envía de vuelta al cliente. Esta información es valiosa, debido a que revela si la petición fue respondida con éxito por el servidor (los códigos que empiezan por 2),

una redirección (los códigos que empiezan por 3), un error provocado por el cliente (los códigos que empiezan por 4), o un error en el servidor (los códigos que empiezan por 5).

2326 (%b): Indica el tamaño del objeto retornado por el cliente, no incluidas las cabeceras de respuesta. Si no se respondió con ningún contenido al cliente, este valor mostrará valor "-".

"http://www.example.com/start.html" (\"%{Referer}i\"): La cabecera de petición de *HTTP "Referer",* muestra el servidor del que proviene el cliente. (Esta debería ser la página que contiene un enlace o que contiene a */apache_pb.gif*).

"Mozilla/4.08 [en] (Win98; I ;Nav)" (\"%{User-agent}i\"): La cabecera de petición *HTTP "User-Agent"*. Es la información de identificación que el navegador del cliente incluye sobre sí mismo.

En el Departamento de Componentes del centro TLM los administradores del servicio web Apache acceden con frecuencia a los log de acceso para visualizar los registros de acceso que contienen los mismos con el objetivo de conocer las características de las PC clientes que acceden a los servicios, incluyendo la dirección del servicio al que accedió. Otras de las tareas frecuentes de los administradores es la de acceder al archivo principal apache2.conf y realizar modificaciones para la configuración del servicio web Apache.

1.5. Herramientas informáticas para la configuración del servicio web Apache

Se realiza el análisis de sistemas relacionados al campo de acción de la investigación sobre la configuración de servicios web Apache, con el objetivo de asimilar los elementos característicos y funcionales que puedan aportar al desarrollo de la propuesta de solución.

1.5.1. Internacional

Webmin

Webmin, es una aplicación web desarrollada por Jamie Cameron y la Comunidad Webmin para la administración de sistemas operativos con distribuciones GNU/Linux de forma local y remota (ver figura 1). Este sistema permite la configuración de la mayoría de los servicios mediante módulos entre los cuales se encuentra: Apache *WebServer*, *BIND DNS Server, CVS Server, DHCP Server, FTP, Proxy y Jabber*. Incluye un servidor web que funciona con SSL (protocolo *HTTPS*). Está construido a partir de módulos, los cuales tienen una interfaz a los archivos de configuración, lo que permite añadir nuevas funcionalidades (Webmin 2016).

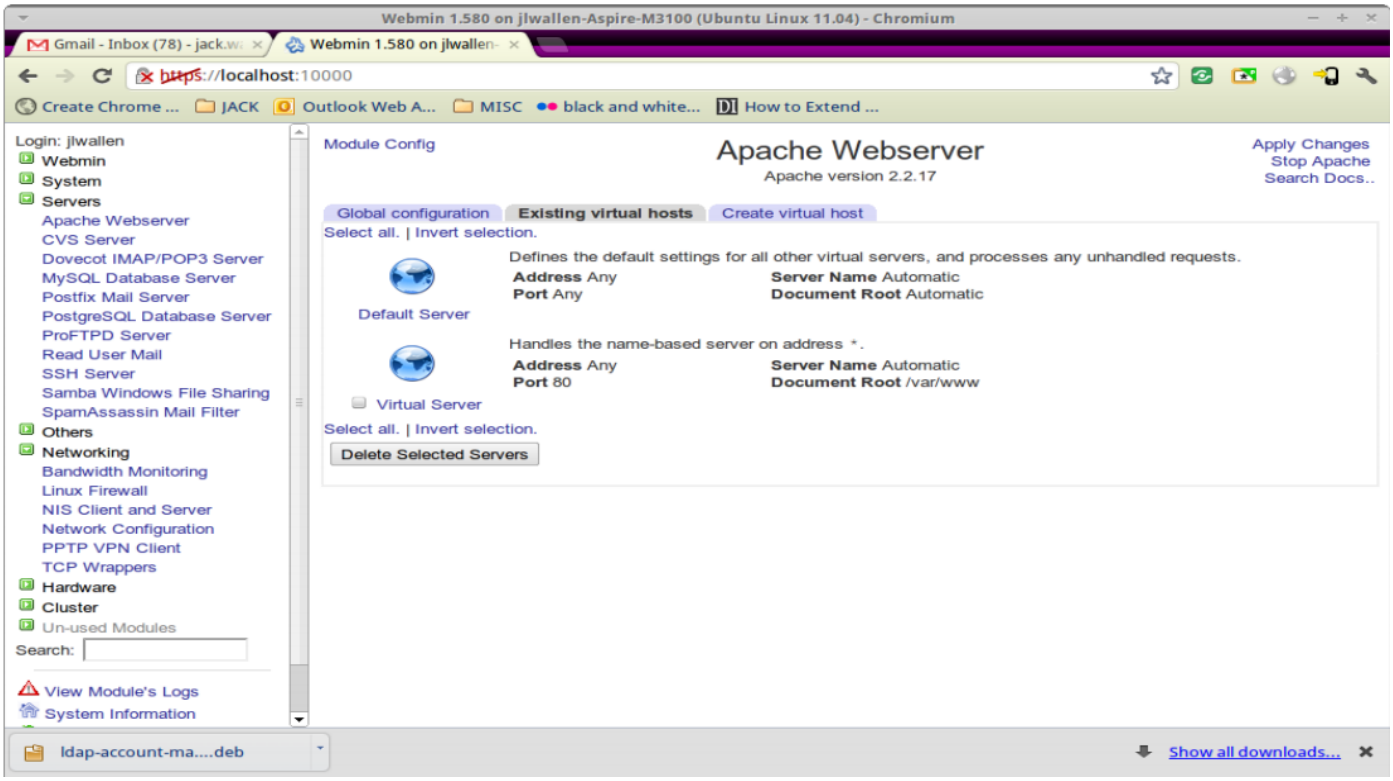


Figura 1: Interfaz del Webmin. Fuente: Sitio oficial de Webmin.

Apache GUI

Apache GUI, como se muestra en la figura 2, es una herramienta web gratuita y de código abierto que está diseñada para ayudar a administrar el servidor web Apache. Posibilita administrar los hosts virtuales, instalar y eliminar módulos de Apache y abrir varias pestañas para editar múltiples archivos de configuración al mismo tiempo. La versión más actual de Apache GUI es compatible con los sistemas operativos *Solaris 10, Ubuntu, Debian, Fedora, Cent OS, RHEL, Open SUSE, MAC OSX y Windows* (ApacheGUI 2015) .

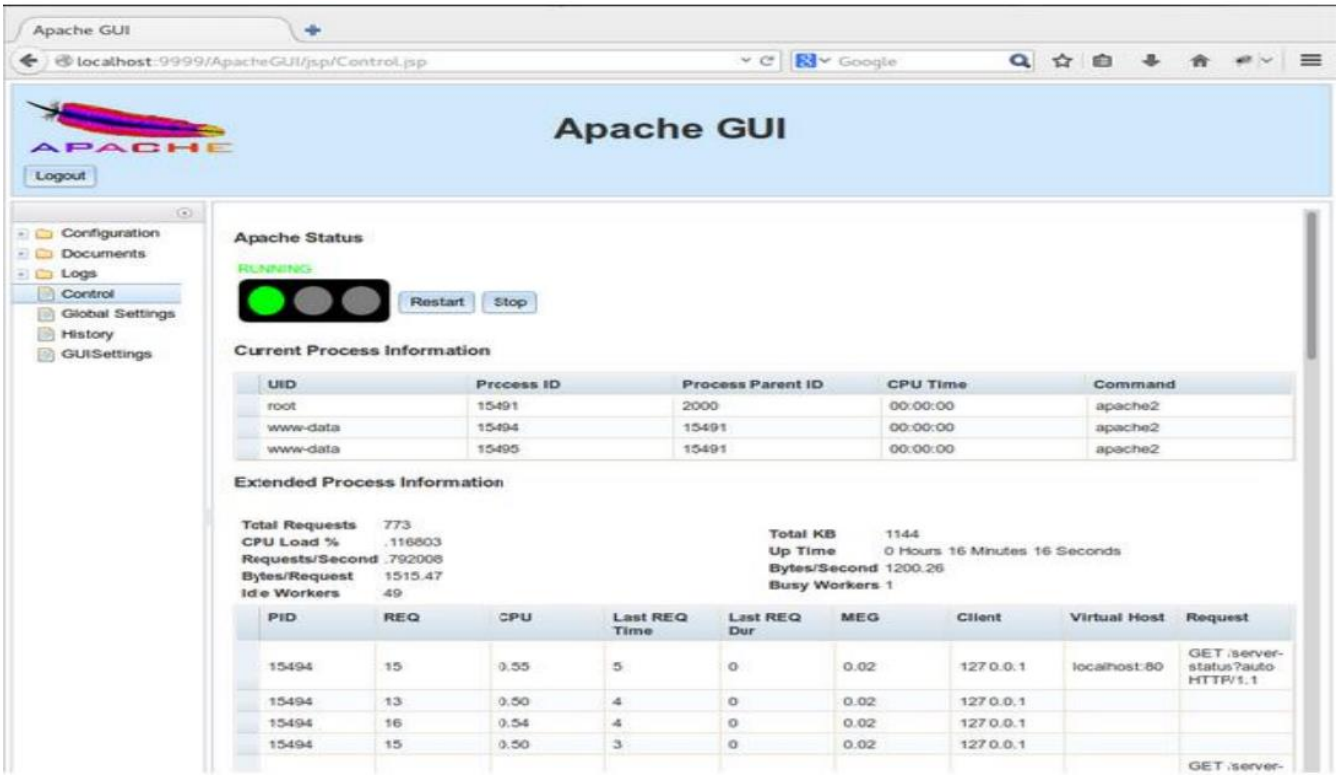


Figura 2: Interfaz de Apache GUI. Fuente: Sitio oficial de Apacge GUI.

1.5.2. Nacional

HMAST

La Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST) permite la administración de servicios telemáticos con la distribución cubana *GNU/Linux Nova*, siendo esta la que se emplea en los procesos de migración. HMAST no introduce elementos (procesos, programas, ficheros, base de datos) ajenos o innecesarios para el funcionamiento del servicio telemático en los servidores de manera que se optimice el uso de los recursos en el servidor. HMAST (ver figura 3), puede ser utilizada tanto por usuarios avanzados como por aquellos que poseen conocimientos básicos asociados a los servicios telemáticos en *GNU/Linux.*

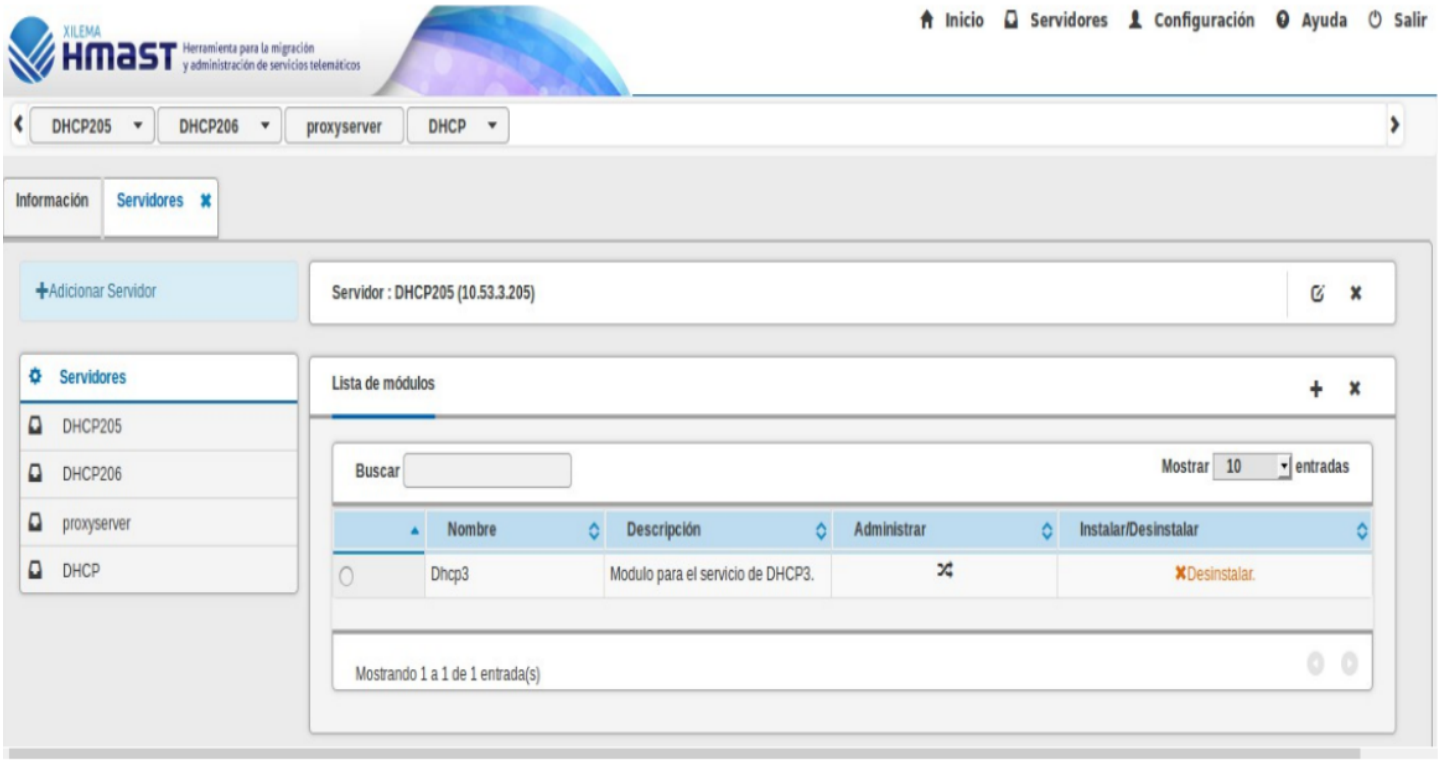


Figura 3: Interfaz de HMAST. Fuente: Sitio oficial de HMAST.

GAMSW

GAMSW (Gestión, Administración y Monitorización de Servidores Web), es un sistema para la gestión, administración y monitoreo de servidores web Apache en el Departamento de Componentes del Centro Telemática de la Facultad 2 de la UCI (ver figura 4). Fue desarrollado como parte de un trabajo de diploma durante el curso 2017-2018 (González 2018). El mismo permite administrar Apache en diferentes computadoras servidoras de manera remota, además de establecer las conexiones de manera segura a través del protocolo de seguridad SSH (*Secure SHell*) y brindar una transmisión segura de la información a través del protocolo SSL (*Secure Sockets Layers*). Otra de sus funciones es la de gestionar host virtuales y mostrar de manera estructurada el almacenamiento lógico de los mismos.

El sistema fue desarrollado para usuarios con rol de administrador y cuenta con un total de 13 funcionalidades. Para su desarrollo fueron utilizadas las herramientas informáticas y lenguajes de programación siguientes: Python en su versión 3.6.0 como lenguaje utilizado del lado del servidor. Del lado del cliente fueron utilizados HTML 5, CSS 3 y JavaScript. Django en su versión 1.10.8 fue utilizado como

marco de desarrollo, así como la librería jQuery 1.9.1 y el marco Bootstrap 3.3.1. Para el análisis y diseño orientado a objetos fue utilizada la herramienta informática Visual Paradigm en su versión 8.0 y PyCharm 2017.3.3 para el entorno de desarrollo integrado. El sistema gestor de base de datos utilizado fue el SQlite 3.

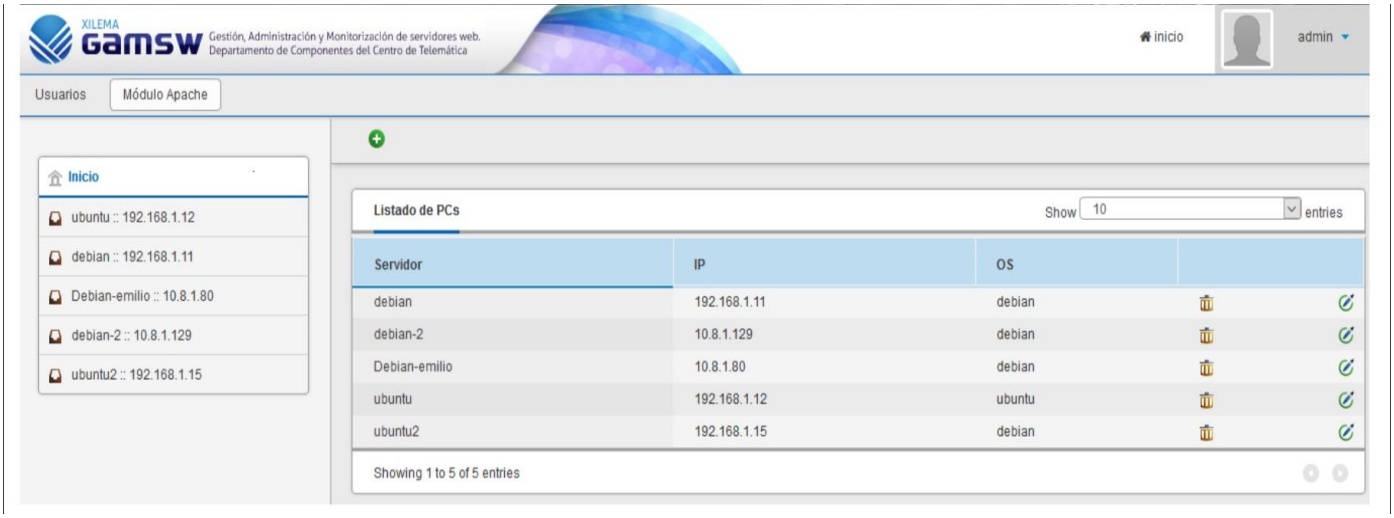


Figura 4: Interfaz de GAMSW. Fuente: Sistema GAMSW 1.0.

1.6. Conclusiones del análisis realizado

Con el análisis realizado se constató que los sistemas antes descritos aportan características útiles para el desarrollo de las nuevas funcionalidades para el sistema GAMSW que darán solución a las necesidades identificadas en el Departamento de Componentes del Centro TLM presentes en la problemática. La herramienta Webmin tiene un aporte significativo para el desarrollo de la solución ya que permite configurar casi todas las características de Apache, lo que ayudará a la implementación de las configuraciones dentro del archivo principal de Apache. El sistema Apache GUI permite validar configuraciones, característica que será aprovechada para validar, en la propuesta de solución, las configuraciones realizadas a los parámetros dentro del archivo principal. Otra característica de interés fue encontrada en la herramienta HMAST, al introducir errores en este sistema, los cambios no son guardados en el servidor y se le notifica al usuario para que sean corregidos, lo que favorecerá a la solución en la detección de errores y el no guardado de los mismos a la hora de modificar el archivo apache2.conf del servicio Apache. El sistema GAMSW sirve como base para el desarrollo de la propuesta de solución ya que forma parte de los productos del centro de TLM. Entre las características de este sistema que serán de utilidad se encuentran: instalar el servicio web Apache en computadoras remotas, así como desinstalarlo, iniciarlo, detenerlo, pausarlo y ver su estado, lo

que permitirá el ahorro de la realización de estos procesos para comprobar el funcionamiento de las nuevas funcionalidades. El código usado para los permisos de acceso a las PC del tipo SSH y los permisos root de escrituras en archivos serán reutilizados en la implementación de las funcionalidades mostrar información de log, configurar archivo principal y mostrar consola de administración.

1.7. Metodología de desarrollo

Las Metodologías de Desarrollo de *Software* surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto. Su principal función es guiar a los desarrolladores al crear un nuevo proyecto, pero los requisitos de un *software* a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del mismo (Carrillo Perez 2008).

1.8. Proceso Unificado Ágil (AUP-UCI)

La metodología de desarrollo de *software Agil Unified Process (*por sus siglas *AUP)* para la UCI, es una variación de la metodología ágil AUP creada por la universidad.

Fases Variación AUP-UCI (Rodríguez 2014)**:**

1. **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

2. **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del *software*.

3. **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Escenario para la disciplina Requisitos:

Para la realización de este trabajo se utilizará el escenario No 4 de la metodología AUP_UCI debido a las características que presenta el sistema a implementar. El mismo plantea que en los proyectos que no modelen el negocio solo se puede modelar el sistema con Historias de Usuarios (HU).

1.9. Herramientas informáticas, lenguajes y tecnologías utilizadas

Las herramienta, leguajes y tecnologías empleados se corresponeden con los utilizados para el desarrollo del sistema. A continuación se describen los mismos.Para la realización de todo sistema informático se hace uso de las herramientas y tecnologías existentes. La correcta selección de la misma va en dependencia de la calidad del producto final, del equipo de desarrollo, o del propio sistema en sí.

1.10. Lenguaje de programación

Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90. Es un lenguaje potente y fácil de aprender. Tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos, es construido sobre objetos que combinan datos y funcionalidades. La sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hacen de éste un lenguaje ideal para el desarrollo rápido de aplicaciones, en muchas áreas y en la mayoría de las plataformas (Rossum 2017).

Se decide utilizar Python 3.6.0 porque facilita el trabajo con expresiones, clases definidas por el usuario, instrucciones y tipos de datos simples. Permite separar el sistema en módulos, propiciando una mayor organización del trabajo realizado. Promueve un estilo de codificación fácil de leer y visualmente agradable.

HTML

HTML (*HyperText Markup Language)* es un lenguaje sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada, con enlaces que conducen a otros documentos o fuentes de información relacionadas. Es el lenguaje con el que se define el contenido de las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes y listas (Gauchat 2012).

Se decide utilizar HTML5 debido a que incorpora nuevos elementos que ayudan a identificar cada sección del documento y organizar el cuerpo del mismo, por lo que las secciones más importantes son diferenciadas y la estructura principal ya no depende de los elementos como tablas y etiquetas div.

CSS

CSS (*Cascade Style Sheets)*, también llamado hojas de estilo en cascada, es un lenguaje de marcado que se emplea para dar formato a un sitio web. Es decir, funciona en conjunto con los archivos HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes entre otros. Permite separar la estructura de la presentación del documento (Gauchat 2012).

Se decide utilizar CSS 3 para la aplicación de estilos que se muestran en el sistema a desarrollar porque permite organizar el documento principal. Permite incrementar la velocidad de carga y aprovechar las nuevas características de HTML5, propiciando un entorno del sistema más atractivo y amigable para los usuarios.

JavaScript

Es un lenguaje de programación interpretado, dialecto del estándar *ECMAScript* (Asociación Europea de Fabricantes de Computadoras). Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Javascript es soportado por la mayoría de los navegadores como *Internet Explorer, Netscape, Opera y Mozilla Firefox*. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como aparición y desaparición de texto, acciones que se activan al pulsar botones u otros elementos y ventanas con mensajes de aviso al usuario (Castillo 2017).

Se decide utilizar JavaScript para el desarrollo de la solución propuesta porque permite controlar las ventanas del navegador y el contenido que muestran. Permite comprobar los datos que el usuario introduce en un formulario antes de enviarlos, característica que favorecerá a la funcionalidad configurar archivo principal.

1.11. Herramientas

Marco de desarrollo: Django

Django es un marco de desarrollo en Python de código abierto, legible que permite la creación rápida de páginas y aplicaciones web. Utiliza una variación de la arquitectura MVC (Modelo-Vista-Controlador) llamada MTV (Modelo-Plantilla-Vista). Impulsa el desarrollo de código limpio al promover buenas prácticas de desarrollo web. Contiene su propio ORM (mapeo-objeto-relacional), una capa intermedia entre el código y la base de datos que separa la integración entre ambos por completo. Ofrece una serie de soluciones a problemas comunes, basándose en el uso de patrones de diseño (Django 2016).

Por lo anteriormente expuesto se decide utilizar Django 1.11 como marco de desarrollo para la implementación de la solución propuesta.

jQuery

jQuery es una librería JavaScript *de* código abierto, que funciona en múltiples navegadores, y que es compatible con CSS3. Su objetivo principal es hacer la programación mucho más fácil y rápida del lado del cliente. Integra funcionalidades para trabajar con AJAX (*Asynchronous JavaScript And XML,* por sus siglas en inglés) o JavaScript asíncrono y XML. Provee de un mecanismo para la captura de eventos (Castillo 2017).

Se decide utilizar JQuery en su versión 1.9.1 para el desarrollo del sistema propuesto porque facilita la selección de elementos HTML. Permite cambiar la información de una página web sin necesidad de recargarla. Provee soporte para viejos navegadores y puede ser utilizado como una forma simple de reemplazar funciones de HTML5.

Bootstrap

Bootstrap, es un marco que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Es una herramienta ágil para construir sitios web (Bootstrap 2015).

Se escoge Bootstrap en su versión 3.3.1 para el desarrollo de la solución propuesta debido a que se integra con las librerías JavaScript. Permite la realización de formularios, menús, botones, la utilización de íconos y otros componentes para el desarrollo de las aplicaciones web.

Herramienta de Modelado

Visual Paradigm

Es una herramienta CASE (*Computer Aided Software Engineering*) o Ingeniería de *Software* Asistida por Computadora. Visual Paradigm 8.0 puede ayudar en todos los aspectos del ciclo de vida de desarrollo del *software*: análisis y diseño orientado a objetos. Posibilita el desarrollo de la ingeniería inversa y directa. Se encuentra disponible en múltiples plataformas como Linux y Microsoft Windows. Permite realizar diagramas para el diseño enfocado a los procesos del negocio y exportar e importar componentes como imágenes XML (Paradigm 2014).

Se selecciona Visual Paradigm 8.0 para apoyar el modelado de los diagramas ya que varias versiones de UML y permite modelar las clases del sistema de manera visual.

Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es una aplicación que ofrece amplias facilidades para un equipo de programadores durante el desarrollo de *software*. Un IDE normalmente consiste en un editor de código fuente, herramientas de construcción automáticas y un depurador. Varios IDEs modernos se integran con características de codificación inteligentes (IDE 2015). Uno de los objetivos del IDE es reducir la configuración necesaria para juntar múltiples utilidades de desarrollo, y ofrecer el mismo conjunto de capacidades como una unidad coherente.

PyCharm

PyCharm es desarrollado por la empresa JetBrains. Es un entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica y soporte para el desarrollo web con Django. Soporta varias bases de datos, entornos virtuales e intérpretes de Python 2.x, 3.x (JetBRAINS 2017).

Se decide utilizar PyCharm 2017 3.3 como entorno de desarrollo integrado porque permite la integración con marco web como Django. Posibilita el autocompletado de código, el resaltado de sintaxis y de errores.

Sistema Gestor de Base de Datos

Es un programa orientado a la gestión y diseño de bases de datos, permitiendo su creación, modificación, atributos e interfaz, interactuar y explotar sus contenidos. El objetivo principal de un SGBD (Sistema Gestor de Base de Datos) es proporcionar un entorno eficaz a la hora de almacenar y recuperar la información de la base de datos (Cobo 2017). Permite a los usuarios una visión abstracta de la información, es decir, el sistema ahorra al usuario la necesidad de conocer al detalle cómo se almacenan los datos.

SQLite 3

SQLite es una biblioteca en proceso que implementa un motor de base de datos SQL transaccional independiente, sin servidor y de configuración cero. SQLite lee y escribe directamente en archivos de disco ordinarios. Una base de datos SQL completa con múltiples tablas, índices, disparadores y vistas. Está contenida en un solo archivo de disco. El formato de archivo de la base de datos es multiplataforma: puede copiar libremente una base de datos entre sistemas de 32 bits y de 64 bits (SQlite 2018).

Se selecciona SQlite 3 como gestor de bases de datos debido a que viene integrado a Python, sus bases de datos pueden ser portadas sin ninguna configuración o administración. Cuenta con diferentes interfaces API (Interfaces de Programación de Aplicaciones), las cuales permiten trabajar con Python.

1.12. Conclusiones parciales

En este capítulo se abordaron los referentes teóricos de la configuración de los servicios web Apache. El estudio del estado actual de los sistemas para la configuración de estos servicios permitió constatar que ninguno constituye una solución al problema planteado en esta investigación, por lo que se decide evolucionar el sistema GAMSW del departamento de Componentes del Centro Telemática.

Se seleccionó la metodología AUP UCI para guiar el desarrollo, especificando el uso del escenario 4 para la disciplina de requisitos. Además, se definieron las herramientas y lenguajes informáticos a emplear, teniendo en cuenta sus características para contribuir al desarrollo de la propuesta de solución.

Capítulo 2. Sistema de configuración de servidores web Apache

Introducción

En el presente capítulo se describen los requisitos funcionales y no funcionales de la propuesta de solución, la arquitectura seleccionada, así como los artefactos generados en la etapa correspondiente al modelado del sistema. Así mismo, se describen los patrones arquitectónicos y de diseño a tener en cuenta para el desarrollo, de acuerdo a la propuesta arquitectónica de la versión previa del sistema.

2.1. Propuesta de solución

Con el objetivo de facilitar el proceso de configuración de servicios web Apache en el Departamento de Componentes del centro TLM, se decide evolucionar el sistema GAMSW a su versión 2.0 para dotarlo de funcionalidades que resuelvan las deficiencias planteadas en la problemática, derivadas del uso de la versión anterior. El sistema debe tener en cuenta la búsqueda del log de tipo *acces*, el cual almacena la información asociada al cliente, en cuanto a ruta más visitada, horarios picos de visita, sistema operativo del cliente, la dirección ip y el navegador desde el que accede al sistema. Esta información será mostrada de manera gráfica en una tabla.

Con respecto a la configuración del archivo principal el sistema permitirá al administrador realizar las configuraciones mediante un formulario, el sistema buscará el archivo principal apache2.conf que contiene diversos parámetros, extraerá los parámetros server *MaxKeepAliveRequests, Hostnamelookups, ServerRoot, ErrorLog, LogLevel, Timeout, KeepAlive, KeepAlive Timeout*, los cuales se mostrarán en el formulario como parámetros editables. Luego de editados, el sistema validará los cambios tras la selección de la opción Guardar, y si están correctos procederá a guardarlos en el archivo apache2.conf de Apache.

La propuesta de solución brindará una consola de administración para facilitar el trabajo a los administradores que prefieren trabajar con la misma. La consola permitirá ejecutar los *comandos service apache2 start, service apache2 stop, service apache2 status, service apache2 restart, apt-get install apache2 y apt-get remove apache2* pertenecientes a los procesos que ejecuta la aplicación.

Para garantizar la seguridad del sistema se usó desde su primera versión la autenticación de usuario para poder acceder al mismo. En las funcionalidades agregadas será aprovechado este elemento, implementando cada método de manera que se cumpla que, para su ejecución, el sistema primero verifique

que el usuario solicitante de la petición está o no autenticado en el sistema. En caso positivo, el método será ejecutado, en caso contrario el sistema redirigirá al usuario a la ventana de autenticación para autenticarse. El sistema mantendrá su arquitectura cliente-servidor definida en su primera versión, así como el patrón arquitectónico *Model template View* (MTV por sus siglas en inglés).

2.2. Requisitos de Software

Según (Pressman 2010) los requisitos de *software* constituyen las necesidades de los clientes, las funcionalidades y las restricciones que debe cumplir el sistema *software*. Los mismos se clasifican en funcionales y no funcionales.

2.3. Requisitos Funcionales

Los Requisitos Funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los RF de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Vila Alvarez 2018). A continuación, se especifican los RF de la aplicación a desarrollar.

RF 1. Mostrar información de log: permite al administrador visualizar la información de los log de los servidores web Apache.

RF 2. Mostrar consola de administración: permite al administrador visualizar y utilizar la consola de administración.

RF 3. Modificar archivo principal: permite realizar configuraciones en algunos parámetros del archivo principal de apache apache2. conf.

RF 3.1. Mostrar parámetros configurables: muestra una lista con los parámetros que pueden ser configurados por el administrador.

RF 3.2. Modificar parámetro: permite configurar un parámetro seleccionado.

RF 4. Realizar Balance de carga: permite realizar un balanceo de carga entre los servidores web apache.

2.4. Requisitos no Funcionales

Los Requisitos No Funcionales (RNF), como su nombre sugiere, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento (Vila Alvarez 2018). De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. Los RNF rara vez se asocian con características particulares del sistema; más bien, estos requisitos especifican o restringen las propiedades emergentes del sistema.

Para la definición fue utilizada la técnica de la entrevista, que es utilizada para recopilar información del cliente y en la cual influye, en la obtención del resultado final, la predisposición y experiencia de la persona entrevistada. En este caso el cliente cuenta con un alto nivel de experiencia debido a que cumple con el rol de administrador del sistema y posee el título de Ingeniero en ciencias informáticas. Luego de realizada la entrevista se arribó a la conclusión de que el sistema debe cumplir con los siguientes RNF:

RNF 1. Interfaz de usuario: el sistema debe tener indicadores que permitan dar a conocer al usuario las acciones que debe realizar, por ejemplo, botones con íconos sugerentes y alternativa textual.

RNF 2. Usabilidad: las funcionalidades principales del sistema estarán orientadas a íconos para un mayor reconocimiento por parte del usuario.

RNF 3. Seguridad: el sistema controlará el acceso y lo permitirá solamente a usuarios autorizados. Los usuarios deben ingresar al sistema con un nombre de usuario y contraseña

RNF 4. Seguridad: el sistema controlará la confidencialidad e integridad de la información controlando que el usuario que intente la ejecución de algún método esté primero autenticado en el sistema.

RNF5. Disponibilidad: El sistema debe garantizar el acceso a la información en todo momento siempre y cuando los servidores web Apache se encuentren en funcionamiento.

RNF 6. *Software*:
Se requiere para las PC clientes las siguientes condiciones mínimas:
-Tener instalado algún navegador.

Se requiere para los servidores las siguientes condiciones mínimas:

-Sistema Operativo Debian o Ubuntu.
-Servicio web Apache

Para la definición de los requisitos de hardware la aplicación GAMSW 2.0 fue desplegada en una máquina virtual con Sistema Operativo Ubuntu 16.04 a la cual se le fue disminuyendo las propiedades de procesador, memoria RAM y capacidad de disco duro para comparar su rendimiento con distintas propiedades. Mediante esta prueba se llega a la conclusión de que:

RNF 7. Hardware:
Se requiere para las PC clientes las siguientes condiciones mínimas:
- Procesador: 500MHz.
- Memoria RAM: 512MB.
-Poseer tarjeta de red.

Se requiere para los servidores las siguientes condiciones mínimas:
- Procesador: 2Ghz.
- Memoria RAM: 2GB.
- Disco Duro: 500MB.

2.5. Descripción del Sistema

Un sistema de información basa la parte fundamental de su procesamiento en el empleo de la computación y utiliza dispositivos que se usan para programar y almacenar programas y datos. A continuación, se describen las características con las que cumple el sistema GAMSW 2.0.

2.5.1. Descripción de los actores que interactúan con el sistema

Un actor es un usuario del sistema, esto incluye usuarios humanos y otros sistemas computacionales. El conjunto de funcionalidades a las que un actor tiene acceso define un rol en el sistema y el alcance de su acción (Vila Alvarez 2018) La tabla 1 describe el actor que va a interactuar con el sistema a desarrollar, además se especifica el rol que ocupa dentro del mismo.

Tabla 1: Descripción de los actores del sistema. Fuente: elaboración propia.

Actor	Descripción

Administrador del Sistema	Representa a los especialistas que van a administrar el sistema.

2.5.2. Descripción de los Requisitos Funcionales

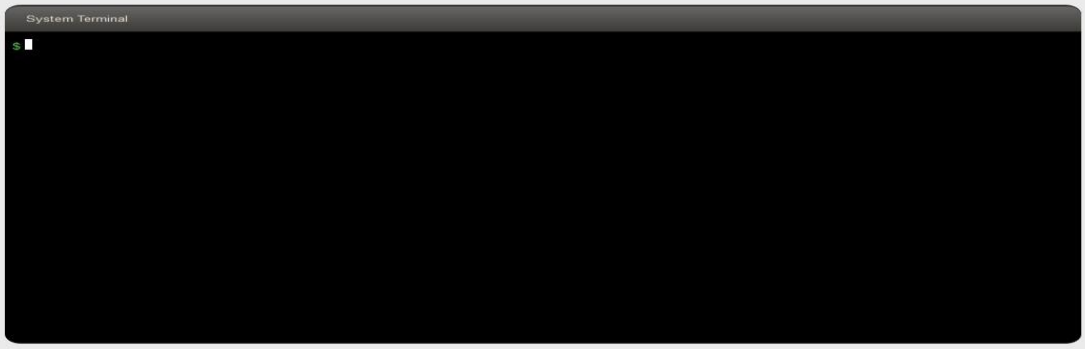
Los Requisitos Funcionales son la entrada esencial para realizar el análisis, diseño, implementación y pruebas del sistema. La propuesta de solución presenta un total de 6 requisitos funcionales. Las tablas 2 y 3 describen los requisitos funcionales números 1 y 2, el resto puede ser verificado en el Anexo 1.

Tabla 2: Descripción del RF 1 Mostrar Información de Log. Fuente: elaboración propia.

Historia de Usuario	
Número: 1	**Nombre del Requisito:** Mostrar Información de Log
Prioridad: Media	
Riesgo en Desarrollo: Afectaciones al personal de trabajo debido a orientaciones de la dirección o la universidad, enfermedad y otros factores.	
Descripción: ✓ El sistema accede al servidor a través del protocolo SSH y luego al registro Access.log del directorio de Apache */var/log/apache2.* ✓ Al acceder al registro busca la información que contiene y muestra a los administradores un listado con los parámetros: rutas visitadas, IP de los visitantes, horarios de visitas, navegadores y sistemas operativos con los que se acceden al sistema.	
Observaciones: ✓ Los parámetros serán mostrados como parte del proceso de análisis de log y los mismos no serán modificables.	
Prototipo de Interfaz	

Tabla 3: Descripción del RF 2. Fuente: elaboración propia.

Historia de Usuario	
Número: 2	**Nombre del Requisito:** Mostrar Consola de Administración
Prioridad: Media	
Riesgo en Desarrollo: Afectaciones al personal de trabajo debido a orientaciones de la dirección o la universidad, enfermedad y otros factores.	
Descripción: ✓ Muestra al administrador una consola de administración mediante la cual podrá realizar configuraciones a los servicios web Apache desde la misma aplicación. ✓ Las configuraciones permitidas son*: service apache2 start, service apache2 stop, service apache2 status, service apache2 restart, apt-get install apache2, apt-get remove apache2.* ✓ La consola recibe un comando y lo ejecuta desde el view.py mediante el uso de la librería OS.	
Observaciones: ✓ Su uso no es de carácter obligatorio, se presenta para los administradores más diestros en comandos por consola.	
Prototipo de Interfaz	



2.5.3. Arquitectura de Software

La arquitectura de *software* es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (Vila Alvarez 2018).

2.5.3.1. Patrones arquitectónicos

Un patrón arquitectónico provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. Estos patrones son plantillas para arquitecturas de *software* concretas, que especifican las propiedades estructurales de una ampliación con amplitud de todo el sistema y tienen un impacto en la arquitectura de subsistemas (Vila Alvarez 2018).

La selección de los patrones arquitectónicos es una decisión fundamental en el desarrollo de un *software* pues permiten darle solución a un problema en específico. En la solución de la investigación se mantiene el uso de la Arquitectura Cliente Servidor definida en su primera versión.

Arquitectura Cliente Servidor:

Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes. Normalmente el servidor es una máquina potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos (SGBD). Por otro lado, los clientes

suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red (Vila Alvarez 2018).

Entre las características básicas de una arquitectura Cliente/Servidor se encuentran la combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de *software* que maneja recursos compartidos tales como bases de datos e impresoras. Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, capacidades del disco y dispositivos de entrada y salida. Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo, pues esperan las peticiones de los clientes (Contreras Ortiz 2017).

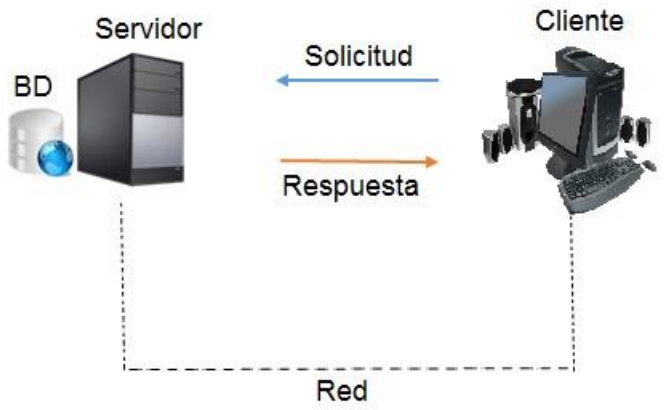


Figura 5: Arquitectura Cliente-Servidor. Fuente: elaboración propia.

Model Template View (MTV)

Para el desarrollo del sistema se decide utilizar el *Model Template View* (MTV por sus siglas en inglés) porque es un estilo de arquitectura de *software* que separa los datos de la aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

El Modelo

El modelo define los datos almacenados. Se encuentra en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, posee métodos también.

Todo esto permite indicar y controlar el comportamiento de los datos (Marrero 2012). En el modelo se representa la clase Servidor.

La Vista

La vista se presenta en forma de funciones en Python, su propósito es determinar qué datos serán visualizados. El ORM (*Object-Relational Mapping*) de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista. La vista también se encarga de tareas conocidas como el envío de correo electrónico, la autenticación con servicios externos y la validación de datos a través de formularios. Lo más importante a entender con respecto a la vista es que no tiene nada que ver con el estilo de presentación de los datos, sólo se encarga de los datos, la presentación es tarea de la plantilla (Marrero 2012).

La Plantilla

La plantilla recibe los datos de la vista y luego los organiza para la presentación al navegador web. Las etiquetas que Django usa para las plantillas permiten que sea flexible para los diseñadores del *frontend*, incluso tiene estructuras de datos como *if*, por si es necesaria una presentación lógica de los datos, estas estructuras son limitadas para evitar un desorden poniendo cualquier tipo de código Python. Esto permite que la lógica del sistema siga permaneciendo en la vista (Marrero 2012). En la plantilla se representan las vistas que mostrarán la interfaz por la cual se administrarán los servicios Apache. El uso de los marcos basados en el patrón de arquitectura MTV permite tener una separación lógica y física de los componentes de la aplicación. Además de contribuir a una elevada organización en el trabajo y que el proyecto sea bien desarrollado.

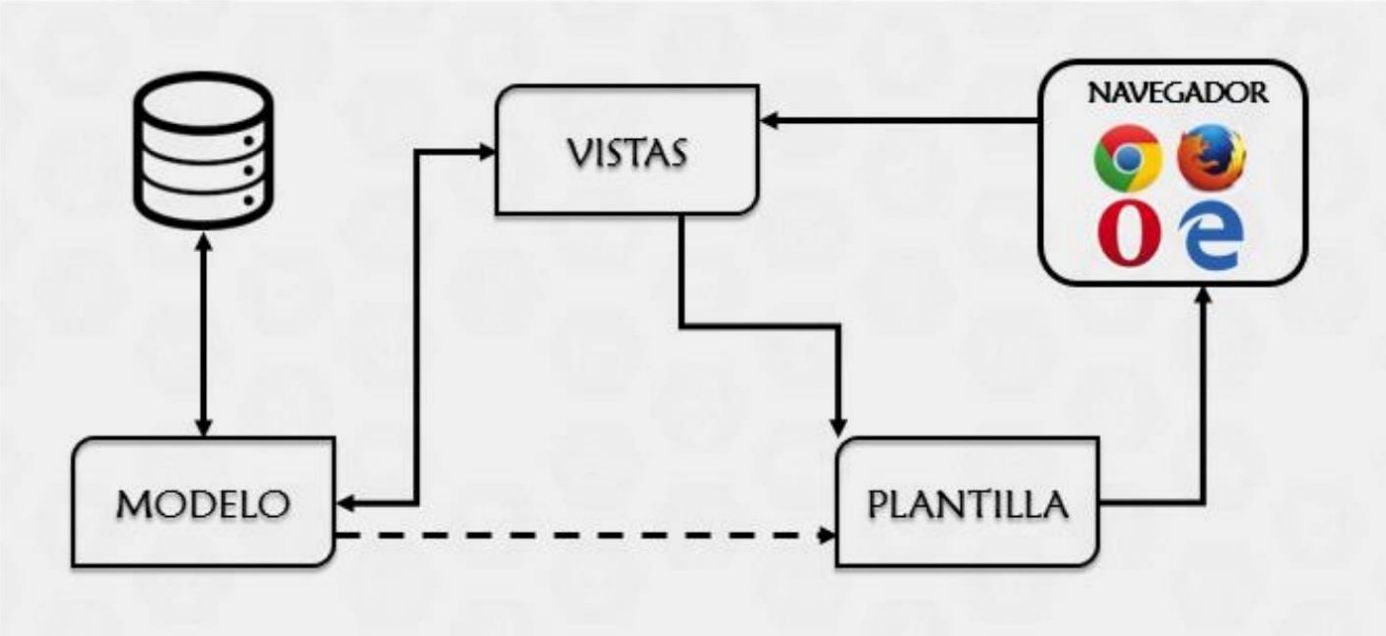


Figura 6: Estilo de arquitectura MTV. Fuente: elaboración propia.

Este estilo de arquitectura se evidencia en el sistema de la siguiente manera:

Modelo

Se evidencia en la clase de python *class EditHttpForm(forms.Form)* la cual hace referencia al formulario que define los datos almacenados a utilizar en la funcionalidad Configurar archivo principal obtenidos del archivo principal *apache2.conf* del servidor web Apache.

La Vista

Se representa en las funciones *def get(self, request)* y *def post(self, request)* del archivo View.py las cuales contienen el código necesario para determinar, dentro de los datos contenidos en el formulario, los parámetros que serán visualizados en el navegador, así como la validación de cada una de estos.

La Plantilla

Se evidencia en la plantilla *editar_http.html* que recibe de la vista los datos a mostrar del formulario. En el mismo se define como se van a organizar y cómo será la visualización de los datos al mostrarse en el navegador web.

2.6. Diagrama de Componentes

Un diagrama de componentes es un diagrama UML compuesto solo por paquetes y las dependencias entre ellos. Un componente es una construcción UML que permite organizar elementos modelo, como casos de uso o clases, en grupos. Los componentes se representan como carpetas de archivos y se pueden aplicar en cualquier diagrama UML. (Pressman 2004). A continuación, se muestran los componentes del sistema GAMSW 2.0.

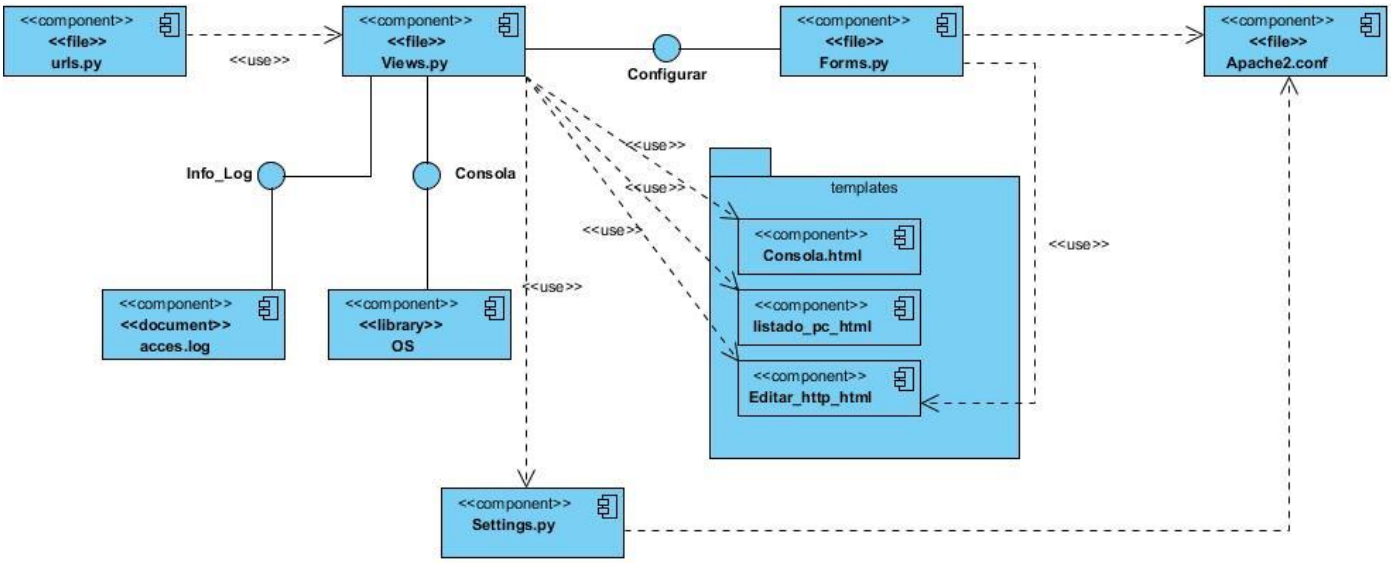


Figura 7: Diagrama de componentes. Fuente: elaboración propia.

Como se evidencia en la figura 7, el archivo *urls.py* usa el archivo *Views.py* el cual a su vez se relaciona, mediante tres vistas de nombre *Info_Log*, Consola y Configurar, al documento *acces.log*, a la librería os y al archivo *Forms.py* respectivamente. El archivo *Views.py* usa el archivo *Settings.py*, así como las vistas *Consola.html*, *listado_pc_html* y *Editar_http_html*, últimos tres también usados por el archivo *Forms.py*. El archivo *Apache2.conf* ubicado en el servidor web Apache es usado directamente por los archivos *Forms.py* y *Settings.py*.

2.7. Modelo de diseño

El modelo de diseño proporciona detalles sobre arquitectura del *software*, estructuras de datos, interfaces y componentes que se necesitan para implementar el sistema. El diseño permite modelar el sistema o

producto que se va a construir. Este modelo se evalúa respecto de la calidad y su mejora antes de generar código; después, se efectúan pruebas y se involucra a muchos usuarios finales. El diseño es el lugar en el que se establece la calidad del *software* (Pressman 2004).

2.7.1. Diagrama de Clases de Diseño

El diagrama de clases del diseño (DCD) muestra las propiedades y funcionalidades de cada clase en el lenguaje de desarrollo y tecnologías seleccionados. Expresa la colaboración y responsabilidades de cada clase entorno al sistema que conforman y muestra cómo quedaría implementada la aplicación en términos lógicos (Pressman 2004). Las figuras 8, 9, 10 y 11 muestran los diagramas de clases del diseño por requisito funcional.

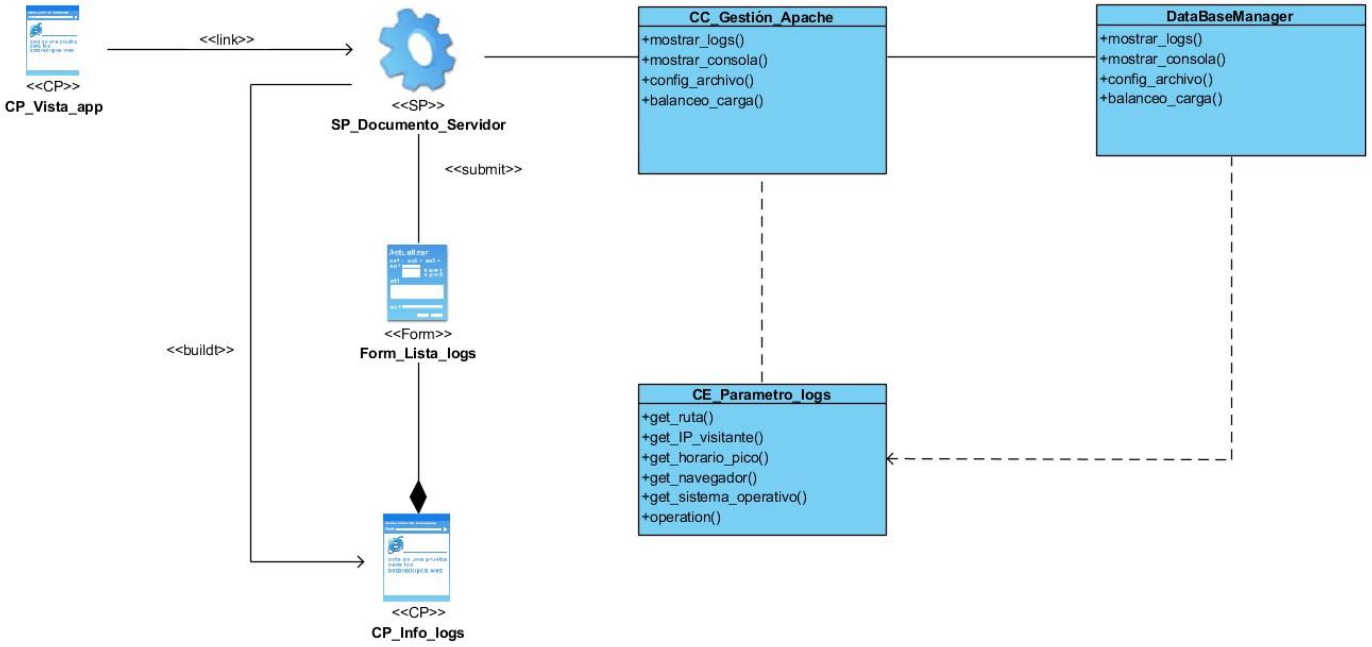


Figura 8: Diagrama de clases de diseño del RF Mostrar información de log. Fuente: elaboración propia.

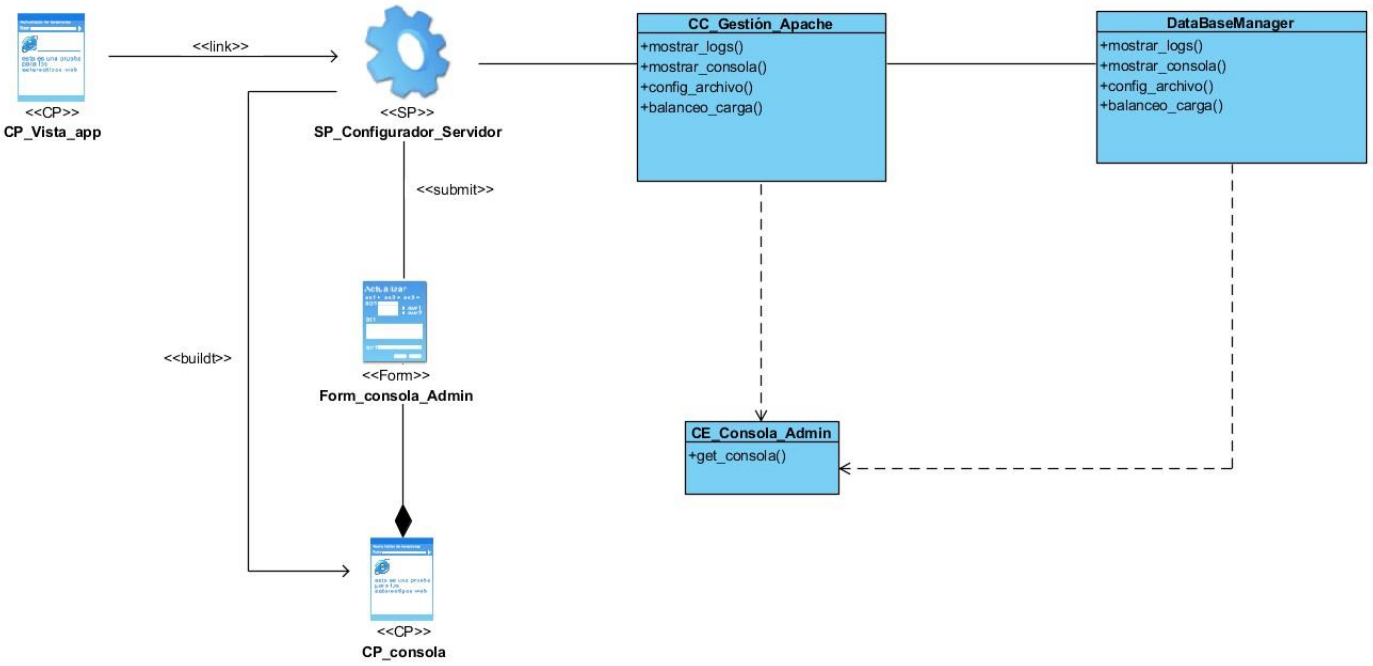


Figura 9: Diagrama de clases de diseño del RF Mostrar Consola de Administración. Fuente: elaboración propia.

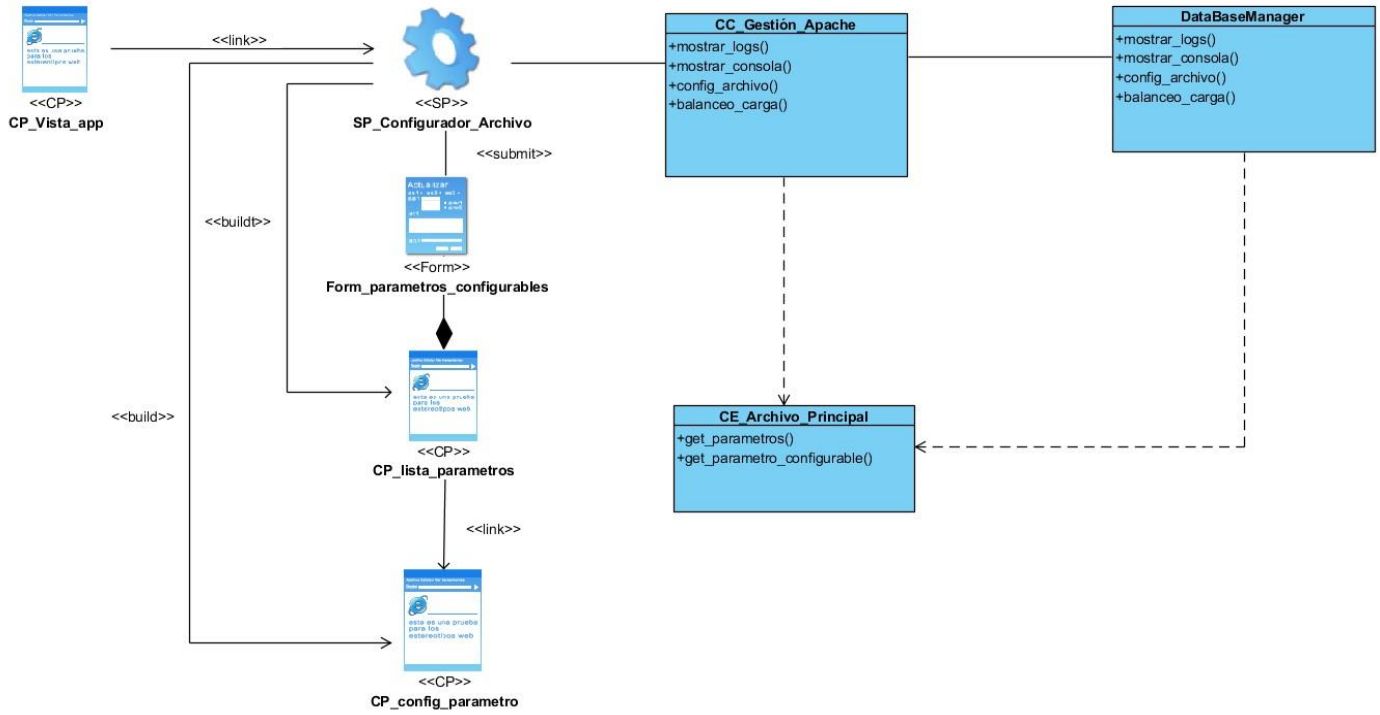


Figura 10: Diagrama de clases de diseño del RF Modificar Archivo Principal. Fuente: elaboración propia.

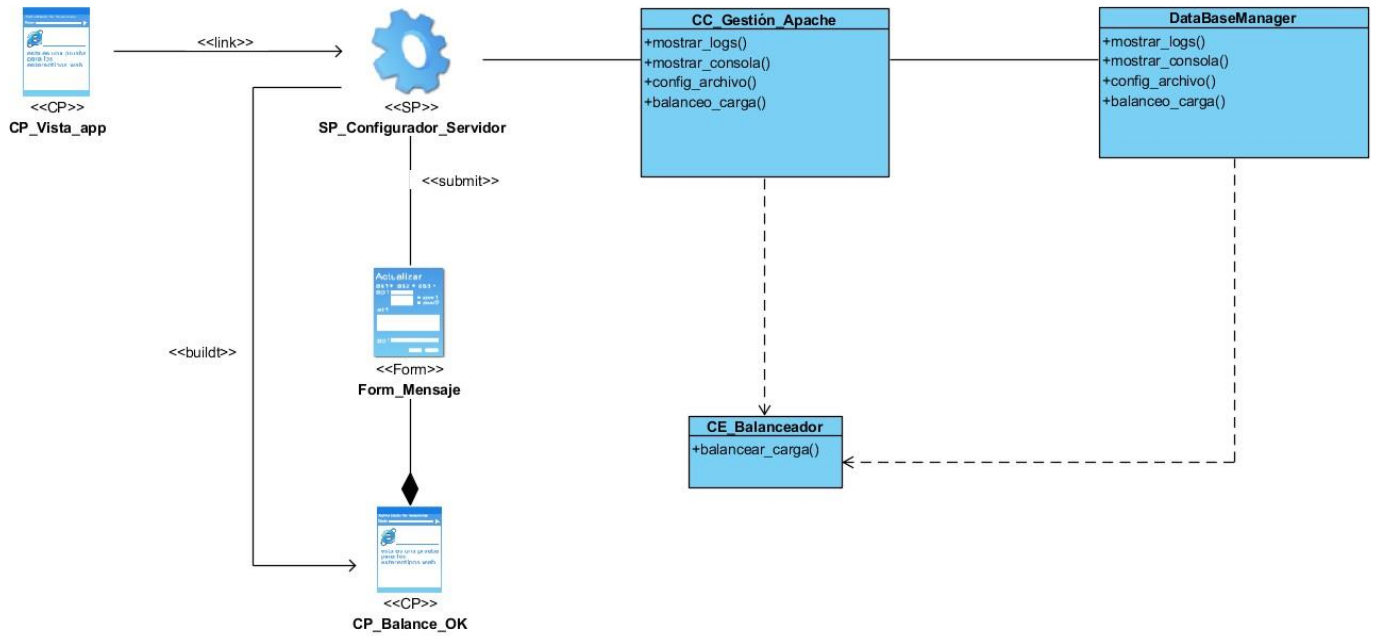


Figura 11: Diagrama de clases de diseño del RF Realizar Balanceo de Carga. Fuente: elaboración propia.

2.7.2. Patrones de diseño

Patrones GRASP

GRASP es un acrónimo de *General Responsibility Assignment Software Patterns* (Patrones Generales de *Software* para Asignar Responsabilidades). Estos patrones describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (Grosso 2011).

Experto: El patrón experto se basa en asignar una responsabilidad al experto en información: la clase que tiene la información necesaria para realizar la responsabilidad. De esta manera se logra un sistema más fácil de entender, mantener y ampliar y existen más oportunidades para utilizar componentes en futuras aplicaciones (Visconti y Astudillo 2017).

El patrón experto es usado en todas las clases ya que cada una posee la información correspondiente para realizar la tarea por la cual fue implementada. Este patrón se evidencia en la clase *CE_Parametro_Log*, la cual contiene todos los parámetros referentes a los log.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es

encontrar un creador que conecte con el objeto producido en cualquier evento. Al seleccionarlo como creador, se da soporte al bajo acoplamiento (Visconti y Astudillo 2017).

Este patrón se pone en evidencia al momento de crear una lista de parámetros contenidos dentro de los log, lo que permite al servidor construir un listado con los parámetros especificados mediante la clase CC_*Gestion_Apache*.

Bajo acoplamiento: El patrón bajo acoplamiento impulsa la asignación de responsabilidades, de manera que su localización no incremente el acoplamiento, hasta un nivel que conlleve a los resultados negativos que puede producir un acoplamiento alto. El bajo acoplamiento soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio. En otras palabras, debe haber poca dependencia entre las clases, de manera que se puedan extraer porciones de código de un modo independiente y reutilizarlas en otro proyecto (Mora 2007).

Se refleja en las interfaces en las cuales se declaran los métodos que mediante Spring se publican como servicios. La utilización de Spring mediante la inyección de dependencia proporciona que las clases estén débilmente acopladas ejemplo *Data_Base_Manager*.

Alta Cohesión: Asigna una responsabilidad de modo que la cohesión siga siendo alta. La alta cohesión permite que la clase que tiene una responsabilidad pueda interactuar con otras clases para poder realizar las tareas encomendadas (Larman 2003).

Este patrón se demuestra en la clase *CE_Parámetros_Archivo que* es la responsable de recoger las configuraciones de archivo para crear el archivo modificado.

2.8. Conclusiones parciales

La definición de los principales artefactos ingenieriles correspondientes a la etapa de análisis y diseño de la solución propiciaron una mejor comprensión de los elementos a tener en cuenta para la implementación. La especificación de los requisitos funcionales y no funcionales del sistema de configuración de servicios web Apache para el departamento de Componentes del Centro Telemática 2.0 permitieron comprobar el comportamiento del negocio, el contexto de la aplicación, la estructura del sistema y sus principales características.

Capítulo 3. Implementación y validación de la propuesta de solución

En este capítulo se describen los elementos necesarios para la implementación de la propuesta de solución, teniendo en cuenta los artefactos generados con la aplicación de la metodología AUP-UCI en el capítulo anterior. Además, se elabora un esquema sobre la distribución del sistema en nodos mediante el diagrama de despliegue. Se muestra la aplicación de los estándares de codificación como buena práctica de programación. Se realiza la validación del sistema a partir de diferentes pruebas.

3.1. Modelo de implementación

El modelo de implementación representa cómo los elementos del modelo de diseño se implementan en términos de componentes. De igual forma describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y la relación existente entre ellos (Ivar Jacobson 2000).

3.2. Diagrama de despliegue.

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del *software* se trazan en esos nodos.

Nodo:

Un Nodo es un elemento de hardware o software. Esto se muestra con la forma de una caja en tres dimensiones, como a continuación.

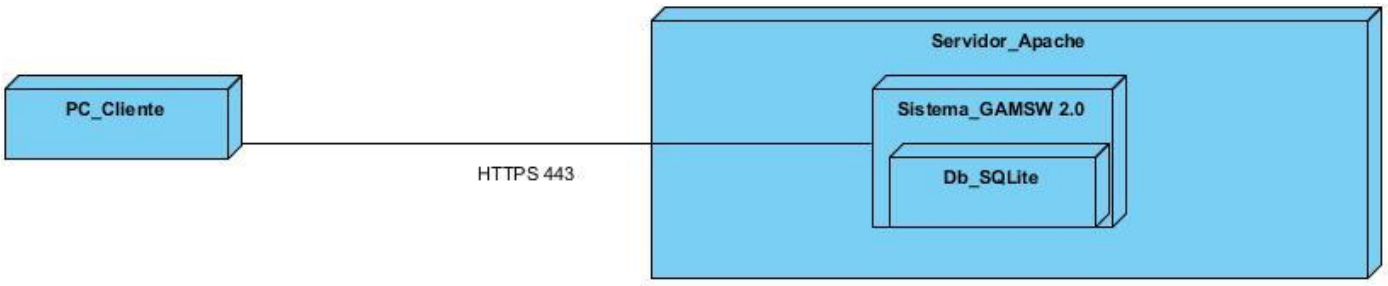


Figura 12: Diagrama de despliegue. Fuente: elaboración propia.

En la figura 12 se evidencia la estructura del despliegue del sistema GAMW 2.0 expuesta mediante la arquitectura en tiempo de ejecución del mismo. Se muestra la *PC_Cliente* quién accederá y usará el sistema GAMSW 2.0, ubicado en el *Servidor_Apache*, mediante el protocolo HTTPS (del inglés *Hypertext Transfer Protocol Secure*), o protocolo de transferencia de hipertexto seguro, utilizando el puerto 443 de TCP (TCP, del inglés *Transmission Control Protocol*), o protocolo de control de transmisión. El sistema accederá a la base de datos *SQlite* de manera local.

3.3. Estándares de codificación

Los estándares de código se refieren a un término que describe convenciones para escribir código fuente en ciertos lenguajes de programación. Los estándares de programación son frecuentemente dependientes del lenguaje de programación que se haya elegido para escribir. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, es de gran importancia para la calidad del *software* y para obtener un buen rendimiento. A continuación, se enumeran los estándares a seguir en la codificación del componente GAMSW 2.0 (Pressman 2010):

Nomenclatura de paquetes:

- Los nombres de todos los paquetes están escritos en minúsculas.
- Los paquetes están definidos como, por ejemplo: os.

Nomenclatura de las clases:

- Comenzar el nombre de la clase en letra inicial mayúscula; en caso de estar compuesto por más de una palabra se utilizan mayúsculas al iniciar cada palabra para distinguirlas.

  Ejemplo:

```
class Login(View):
```

- Los nombres de los métodos de estas clases deben comenzar con letra inicial minúscula; en caso de estar compuesto por más de una palabra se utilizan mayúsculas al iniciar cada palabra para distinguirlas.

  Ejemplo:

```
def start(request):
```

- Los nombres de variables referentes a los metadatos de los documentos deben comenzar con letra inicial minúscula; en caso de estar compuesto por más de una palabra, la inicial de esta debe aparecer en mayúscula.

  Ejemplo:

```
form = EditHttpForm
```

3.4. Pruebas de software

Las pruebas de *software* son importantes porque aseguran el correcto cumplimiento de la funcionalidad del producto, ayudan a ganar confianza, confirman la fiabilidad del uso y previenen defectos en producción, lo cual tiene un impacto económico positivo en la empresa en cuestión. Las pruebas de *software* son una actividad primordial en el proceso de "aseguramiento de la calidad" (Chiu, Alberto y Carbajal 2015).

Realizar pruebas a una aplicación desarrollada es una actividad fundamental que toda empresa productora de *software* debería llevar a cabo. Las pruebas de *software* son un elemento clave en la garantía de la calidad del producto final.

3.4.1. Estrategias de pruebas

Una estrategia de prueba del *software* integra los métodos de diseño de caso de pruebas del *software* en una serie bien planeada de pasos que desembocará en la eficaz construcción del mismo. La estrategia proporciona un mapa que describe los pasos que se darán como parte de la prueba, indica cuándo se planean, cuándo se dan estos pasos, además de cuanto esfuerzo, tiempo y recursos consumirán. Por tanto, cualquier estrategia de prueba debe incorporar la planeación de pruebas, el diseño de casos de pruebas, la ejecución de pruebas, la recolección y evaluación de los datos resultantes (Pressman 2010).

La estrategia de prueba debe indicar los niveles de pruebas (ciclos) que se aplican y la intensidad o profundidad a aplicar para cada nivel de prueba definido.

3.4.2. Niveles de prueba

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes Niveles de Pruebas (Ivar Jacobson 2000):

- Prueba de Unidad: Un componente es la unidad más pequeña especificada de un sistema, las pruebas se llevan a cabo tras la construcción o realización de cada componente para verificar que la implementación se esté llevando conforme a los estándares acordados. El objetivo es comprobar que el sistema, entendido como una unidad funcional, está correctamente codificado.

- Prueba de Sistema: Son las pruebas que se hacen cuando el *software* está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de *software* y *hardware* han sido integrados.

- Prueba de Aceptación: Es la prueba final antes del despliegue del sistema. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

Para realizar las pruebas en el componente GAMSW 2.0 se desarrollarán las pruebas de sistema, las pruebas de unidad y las pruebas de aceptación.

3.4.3. Pruebas de unidad

Se centra en el esfuerzo de verificación de la unidad más pequeña del diseño del *software*, el componente o sistema de *software*. Tomando como guía la descripción del diseño al nivel de componentes, se prueban importantes caminos de control para describir errores dentro de los límites del módulo. El alcance restringido que se ha determinado para las pruebas de unidad limita la relativa complejidad de las pruebas y los errores que estas descubren. Las pruebas se encuentran en la lógica del procedimiento interno y en las estructuras de datos dentro de los límites de un componente. Este tipo de prueba se puede aplicar en paralelo a varios componentes (Pressman 2010). En el contexto de la investigación se aplicará el método de caja blanca.

Método de caja blanca

La prueba de caja blanca, en ocasiones llamada “prueba de caja de vidrio”, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes

para derivar casos de prueba (Pressman 2010). Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:

1. Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.
2. Revisen todas las decisiones lógicas en sus lados verdadero y falso.
3. Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas.
4. Revisen estructuras de datos internas para garantizar su validez.

Para la realización del método de caja blanca se ejecutará la técnica de ruta o trayectoria básica.

Ruta o trayectoria básica

Es una técnica de prueba de caja blanca propuesta por primera vez por Tom McCabe. El método de ruta básica permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba (Pressman 2010).

A continuación, se muestra el caso de prueba correspondiente al método "*def configurar_http(request):* el cual se localiza en la clase "*class Login(View)*:". El objetivo de este método es acceder al archivo principal de Apache apache2.conf y obtener su contenido, lo que posibilitará mostrar los elementos de interés en el formulario de los parámetros a configurar en la funcionalidad Configurar archivo principal. Se selecciona este método teniendo en cuenta la importancia que representa para el resultado de esta funcionalidad. El caso de prueba correspondiente al método *function terminal (caso)* perteneciente a la funcionalidad Mostrar consola de administración, se encuentra en el expediente de tesis, el mismo no es incluido en el documento por el espacio que ocupa en el mismo. El resto de los métodos no se representan debido a que generan un grafo de flujo lineal, por lo que no contienen nodos predicados y el resultado de la complejidad ciclomática es siempre 1. A continuación, se muestra el código fuente referente al método descrito anteriormente y cómo fue aplicada la Técnica de Camino Básico al mismo.

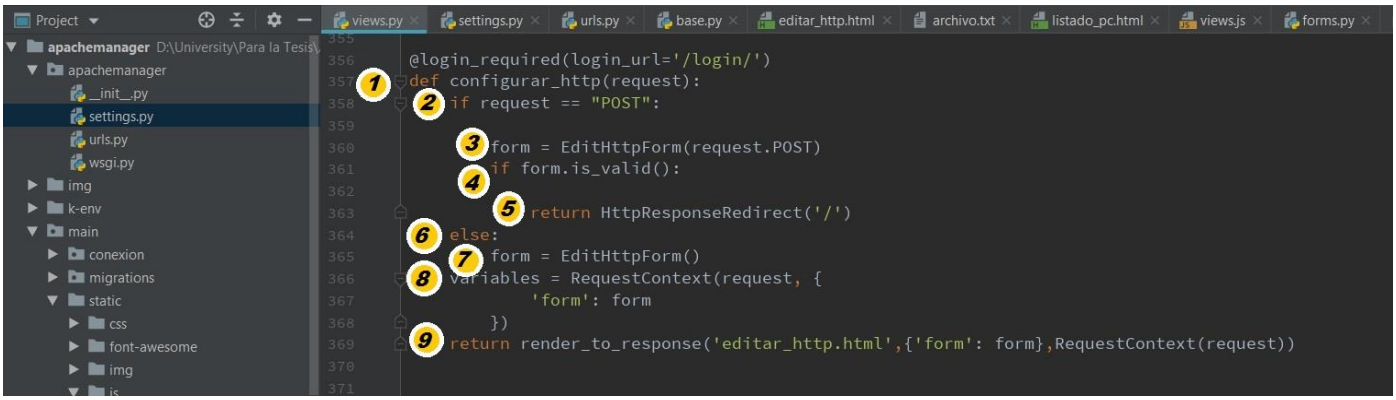


Figura 13: Fragmento de código del método seleccionado. Fuente: elaboración propia.

Antes de considerar el método de ruta básica, debe introducirse una notación simple para la representación del flujo de control, llamado grafo de flujo (o grafo de programa) (Pressman 2010). Cada fragmento de código tiene un número que lo identifica en el grafo de flujo. A continuación, se muestra dicho grafo:

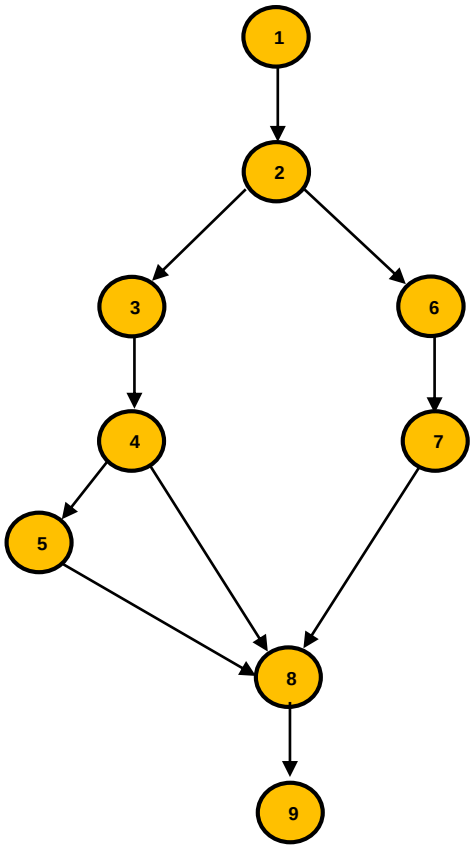


Figura 14: Representación del grafo de flujo de camino básico. Fuente: elaboración propia.

La complejidad ciclomática es una medición de *software* que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba de la ruta básica, el valor calculado por la complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y le brinda una cota superior para el número de pruebas que debe realizar a fin de asegurar que todos los enunciados se ejecutaron al menos una vez. La complejidad ciclomática tiene fundamentos en la teoría de grafo y proporciona una medición de *software* útil. La complejidad se calcula en una de tres formas (Pressman 2010):

1. El número de regiones del grafo de flujo corresponde a la complejidad ciclomática.

2. La complejidad ciclomática *V* (*G*) para un grafo de flujo *G* se define como *V* (*G*)= *E*-*N*+2 donde *E* es el número de aristas del grafo de flujo y *N* el número de nodos del grafo de flujo.

3. La complejidad ciclomática *V*(*G*) para un grafo de flujo *G* también se define como *V (G)= P+1* donde *P* es el número de nodos predicado contenidos en el grafo de flujo *G*.

En el grafo de flujo de la figura 12, la complejidad ciclomática puede calcularse usando cada uno de los algoritmos recién indicados:

1. La cantidad de regiones del grafo de flujo es 3

2. V(G)= 10 aristas – 9 nodos + 2

   V(G)= 3

3. V (G)= 2 nodos predicados+ 1

   V(G)= 3

Por tanto, la complejidad ciclomática del grafo de flujo en la figura 14 es 3.

Una ruta independiente es cualquiera que introduce al menos un nuevo conjunto de enunciados de procesamiento o una nueva condición en el programa. Cuando se establece como un grafo de flujo, una ruta independiente debe moverse a lo largo de al menos una arista que no se haya recorrido antes de definir la ruta (Pressman 2010a). A continuación, se muestran las rutas del grafo de flujo de la figura 14.

Ruta 1*:* 1-2-3-4-5-8-9

Ruta 2*:* 1-2-3-4-8-9

Ruta 3*:* 1-2-6-7-8-9

Luego, se diseñan los casos de prueba que cubren las rutas independientes presentadas, ver tabla 4.

Tabla 4: Caso de prueba de caja blanca para las rutas. Fuente: elaboración propia.

Caso de prueba para la ruta 1	
Entrada	Clic en el botón Guardar o en el botón Archivo Principal
Resultados esperados	Mensaje de error de formulario
Condiciones	El formulario enviado al servidor web Apache para guardar debe presentar error Clic en el botón Guardar
Caso de prueba para la ruta 2	
Entrada	Clic en el botón Guardar o en el botón Archivo Principal
Resultados esperados	Guardado del formulario en el servidor web Apache de manera correcta
Condiciones	El formulario enviado al servidor web Apache para guardar debe estar correcto Clic en el botón Guardar
Caso de prueba para la ruta 3	
Entrada	Clic en el botón Guardar o en el botón Archivo Principal
Resultados esperados	Mostrar en un formulario los parámetros a modificar dentro del archivo principal
Condiciones	Obtener archivo principal guardado en la variable *form* que hace referencia al formulario Clic en el botón Archivo principal

Luego de realizar las pruebas unitarias al componente GAMSW 2.0 empleando la técnica de ruta básica a los métodos mencionados se muestra que el resultado es correcto, por lo que esta prueba concluye de manera satisfactoria.

3.4.4. Prueba de sistema

La prueba de sistema es una serie de diferentes pruebas cuyo propósito principal es ejercitar por completo el sistema basado en computadora. Aunque cada prueba tenga un propósito diferente, todo el sistema funciona para verificar que los elementos se hayan integrado de manera adecuada y que se realicen las funciones asignadas (Pressman 2010).

Prueba de Aceptación

Cada nivel de prueba engloba una técnica de prueba específica según los atributos de calidad que se deseen verificar con las pruebas al *software*. Entre las técnicas de pruebas que se realizan en el sistema está la que evalúa la funcionalidad de este. En esta investigación se realiza el tipo de prueba funcional.

Pruebas funcionales

Este tipo de prueba se enfoca en validar la correcta implementación de las necesidades del cliente. La funcionalidad puede ser vinculada a los datos de entrada y de salida. Los datos de entrada serán ejecutados y mostrarán un resultado y dicho resultado será comparado con el resultado esperado (comportamiento), este proceso se muestra en la figura 15 (Chiu, Alberto y Carbajal 2015).

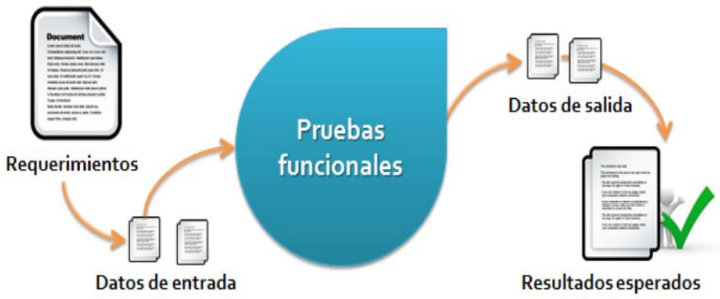


Figura 15: Pruebas funcionales. Fuente: Internet.

Entre las técnicas de pruebas que se realizan en el sistema está la que evalúa la funcionalidad de este. Según los parámetros de evaluación que abarca la técnica se pueden distinguir distintos tipos de pruebas:

- Prueba de Función.
- Prueba de Seguridad.
- Prueba de Volumen.

Al componente GAMSW 2.0 se le aplica la prueba de funcionalidad específicamente la de función.

Pruebas de función (Pressman 2010):

Las pruebas de función aseguran el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados.

Las metas de estas pruebas son:

- Verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio.
- Verificar la apropiada aceptación de datos.

Las pruebas de función están enfocadas en los requisitos funcionales (Casos de Uso) y las reglas del negocio. Estas pruebas utilizan el método de Caja Negra.

El método de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del *software*; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. (Pressman 2010). Estas pruebas pretenden demostrar que las funcionalidades del sistema son operativas, las entradas se aceptan correctamente y que se producen los resultados esperados.

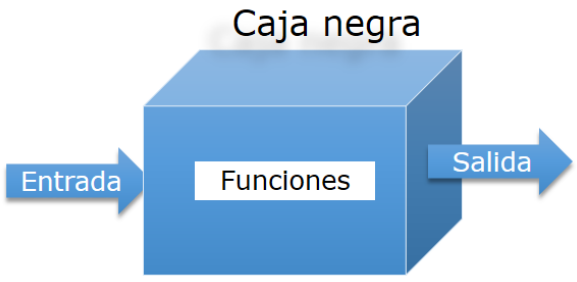


Figura 16: Método de prueba Caja negra. Fuente: elaboración propia.

Para desarrollar las pruebas de caja negra existen varias técnicas. La técnica que se va a utilizar para el componente GAMSW 2.0 es la de partición de equivalencia.

La **partición de equivalencia** divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana. Las clases de equivalencia, pueden desarrollarse y ejecutarse los casos de prueba para cada ítem de datos del dominio de entrada. Los casos de prueba se seleccionan de modo que se revise a la vez el número más grande de atributos de una clase de equivalencia (Pressman 2010).

Los **casos de prueba** deben diseñarse para descubrir errores debido a cálculos erróneos, comparaciones incorrectas o flujo de control inadecuado. Se diseñan para garantizar que: se satisfagan todos los requisitos de funcionamiento, se logran todas las características de comportamiento, todo el contenido es preciso y se presenta de manera adecuada, se logran todos los requisitos de rendimiento, la documentación es correcta y se satisfacen la facilidad de uso y otros requisitos (Vila Alvarez 2018). A continuación se presentan los resultados de aplicar las pruebas para los RF Mostrar consola de administración y Configurar Archivo principal, el requisito Mostrar información de log se encuentra en los anexos (ver Anexo 2).

Tabla 5: Clases de equivalencia. Fuente: elaboración propia.

Identificador	Entrada(clase)	Clases válidas	Clases inválidas
T6-V7	*KeepAlive*	*on-off*	Todo fuera de esas 2 opciones
T6-V8	*KeepAliveTimeout*	número	letras, caracteres especiales
T6-V1	*MaxKeepAliveRequests*	número	letras, caracteres especiales
T6-V4	*ErrorLog*	*/var/log/ErrorLog*	número, caracteres especiales

T6-V6	*Timeout*	número	letras, caracteres especiales
T6-V5	*LogLevel*	*debug,info,notice, warm,error,crit, alert,emerg*	número, caracteres especiales
T6-V3	*ServerRoot*	*/apache2/apache2.conf*	número, caracteres especiales
T6-V2	*HostnameLookups*	*on-off*	Todo fuera de esas 2 opciones
T7-V3	código	*service apache2 start*	Caracteres especiales
T7-V6	código	*service apache2 restart*	Caracteres especiales
T7-V4	código	*service apache2 stop*	Caracteres especiales
T7-V5	código	*service apache2 status*	Caracteres especiales
T7-V7	código	*apt-get intall apache2*	Caracteres especiales
T7-V8	código	*apt-get remove apache2*	Caracteres especiales

En la siguiente tabla se hace referencia a las variables como: V1 (*MaxKeepAliveRequests*), V2 (*Hostnamelookups*), V3 (*ServerRoot*), V4 (*ErrorLog*), V5 (*LogLevel*), V6 (*Timeout*), V7 (*KeepAlive*), V8 (*KeepAlive Timeout*).

Tabla 6: Prueba de caja negra del RF Configurar archivo principal. Fuente: elaboración propia.

Escenario	**Descripción**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Mostrar formulario	Permite mostrar el formulario con los	V	V	V	V	V	V	V	V	Muestra el formulario con los parámetros a configurar	1. Selecciona el archivo principal apache2. conf. 2. Selecciona los parámetros a mostrar.

	parámetros a configurar										3-Inserta los parámetros en el formulario. 4-Muestra el formulario
EC 1.2 Configurar parámetros	Se introducen las configuraciones y se guardan en el archivo principal.	V	V	V	V	V	V	V	V	Guarda las configuraciones realizadas en el archivo principal	1-Selecciona el parámetro a configurar. 2-Configura los parámetros en el formulario. 3-Guarda los cambios en el archivo principal.
		I	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje de error	
		N/A	I	N/A	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	I	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	I	N/A	N/A	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	I	N/A	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	N/A	I	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	N/A	N/A	I	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	I	Muestra un mensaje de error	

En la siguiente tabla se hace referencia las variables como: **V1**(consola), **V2**(Estilo gráfico), **V3**(*service apache2 start*), **V4**(*service apache2 stop*), **V5**(*service apache2 status*), **V6**(*service apache2 restart*), **V7**(*apt-get install apache2*), **V8** (*apt-get remove apache2*).

Tabla 7: Prueba de caja negra del RF Mostrar Consola de Administración. Fuente: elaboración propia.

Escenario	Descripción	V1	V2	V3	V4	V5	V6	V7	V8	Respuesta del sistema	Flujo central
EC 1.1 Mostrar la Consola de Administración	Permite mostrar la consola con un estilo gráfico predefinido.	V	V	N/A	N/A	N/A	N/A	N/A	N/A	Muestra la consola con un estilo gráfico.	1. Selecciona la Consola de Administración a mostrar. 2. Selecciona el estilo gráfico con el que se va a mostrar la consola.
		V	I	N/A	N/A	N/A	N/A	N/A	N/A	Muestra la consola sin un estilo gráfico.	
		I	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Muestra la interfaz del sistema vacía.	
EC 1.2 Ejecución de comandos por consola	Se ejecutan comandos desde la consola.	N/A	N/A	V	V	V	V	V	V	Ejecuta los comandos insertados en la consola.	1-Selecciona el código que ejecuta el comando introducido. 2-Ejecuta el código seleccionado.
		N/A	N/A	I	N/A	N/A	N/A	N/A	N/A	Muestra un mensaje de error	

		N/A	N/A	N/A	I	N/A	N/A	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	I	N/A	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	N/A	I	N/A	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	N/A	N/A	I	N/A	Muestra un mensaje de error	
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	I	Muestra un mensaje de error	

Tabla 8: Descripción de las variables. Fuente: elaboración propia.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*KeepAlive*	Parámetro configurable	No	Permite que cada conexión permanezca abierta para manejar múltiples solicitudes del mismo cliente
2	*KeepAliveTimeout*	Parámetro configurable	No	Controla cuánto tiempo debe esperar la siguiente solicitud después de finalizar la última
3	*MaxKeepAliveRequests*	Parámetro configurable	No	Establece el máximo número de peticiones que se pueden realizar en una conexión
4	*ErrorLog*	Parámetro configurable	No	Especifica la ubicación del fichero que contiene el registro de errores.
5	*Timeout*	Parámetro configurable	No	Define, en segundos, el tiempo que el servidor esperará para recibir y enviar peticiones durante la comunicación, tras los cuales el servidor cierra la conexión.

6	*LogLevel*	Parámetro configurable	No	Especifica el tipo de mensajes que se guardarán en el fichero de registro de errores.
7	*ServerRoot*	Parámetro configurable	No	Define el directorio donde se ubica toda la información de configuración y registro que necesita el servidor para su correcto funcionamiento
8	*HostnameLookups*	Parámetro configurable	No	Resuelve automáticamente la dirección IP de cada conexión que pida un documento del servidor.
9	Estilo gráfico	Visualización de la información	No	Hace referencia al estilo gráfico que toma la consola
10	Consola	Campo de inserción de código	No	Se refiere a la consola que será mostrada en el sistema
11	*service apache2 start*	Código	No	Se refiere al código service apache2 start que será ejecutado en la consola
12	*service apache2 restart*	Código	No	Se refiere al código service apache2 restart que será ejecutado en la consola
13	*service apache2 stop*	Código	No	Se refiere al código service apache2 stop que será ejecutado en la consola
14	*service apache2 status*	Código	No	Se refiere al código service apache2 status que será ejecutado en la consola
15	*apt-get intell apache2*	Código	No	Se refiere al código *apt-get intell apache2* que será ejecutado en la consola
16	*apt-get remove apache2*	Código	No	Se refiere al código *apt-get remove apache2* que será ejecutado en la consola

Resultado de las pruebas de sistema del componente GAMSW 2.0

A continuación, se muestran los resultados de aplicar el método de caja negra, donde se ejecutaron un total de 4 iteraciones. En el siguiente gráfico de barras se evidencia, además, el total de no conformidades identificadas por cada iteración. Igualmente se muestra la cantidad de no conformidades que no procedieron.

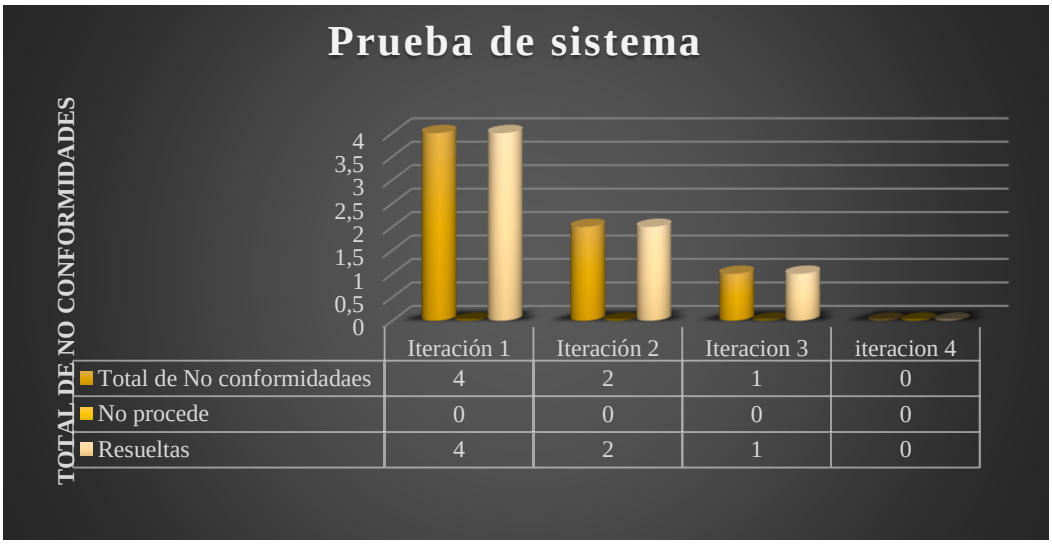


Figura 17: Resultados de las pruebas de caja negra aplicadas al componente GAMSW 2.0. Fuente: elaboración propia.

En la primera iteración se detectaron un total de cuatro no conformidades. De ellas fueron resueltas las cuatro correctamente. En la segunda iteración se detectaron un total de 2 no conformidades, las cuales fueron corregidas con éxito. En una tercera iteración se detectó solo 1 no conformidad y fue resuelta correctamente. En la cuarta iteración no se encontraron no conformidades, validándose de esta manera el correcto funcionamiento del componente GAMSW 2.0 para el Departamento de Componentes del Centro Telemática.

Las no conformidades son los errores que son detectados durante la revisión de la documentación del sistema. Se elabora un documento por cada revisión que se haga y se controlan a través de versiones según se vayan eliminando los errores, hasta que finalmente se hayan erradicado todos los defectos que posea el elemento que se prueba. A continuación, se muestra la tabla donde se hace una clasificación de las no conformidades.

Tabla 9: Clasificación de las No conformidades del componente GAMSW 2.0. ***Fuente: elaboración propia.***

Clasificación	Observaciones
Significativa	Errores en la interpretación de los procesos de la entidad y de funcionalidad.

No significativa	Errores de terminología y de diseño de interfaz.
Recomendación	Errores de redacción y ortografía.

Las no conformidades encontradas en la primera iteración tienen como clasificación:

1. Significativa: en el caso de prueba Mostrar Consola de Administración, describe que al insertar los comandos *service apache2 start y service apache2* stop los mismos no se ejecutan de manera satisfactoria. En el caso de prueba Mostrar Información de log, no se muestran los parámetros rutas más visitadas e ip de los visitantes. En el caso de prueba Configurar archivo principal, la información se guarda con errores debido a que los parámetros no están validados.

2. No significativa: en el caso de prueba Mostrar consola de administración la misma no es mostrada con el estilo gráfico definido.

3. Recomendación: el título “Consola de Administración” de la página principal muestra falta de ortografía.

3.4.5. Pruebas de aceptación del cliente

Cuando se construye un *software* para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al cliente validar todos los requerimientos. Una prueba de aceptación puede variar desde una “prueba de conducción” informal hasta una serie de pruebas planificadas y ejecutadas sistemáticamente. De hecho, la prueba de aceptación puede realizarse durante un período de semanas o meses, y mediante ella descubrir errores acumulados que con el tiempo puedan degradar el sistema. La mayoría de los constructores de productos de *software* usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer solo el usuario final es capaz de encontrar (Pressman 2010).

La prueba alfa se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El *software* se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado.

La prueba beta se realiza en uno o más sitios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. Por tanto, la prueba beta es una aplicación “en vivo” del *software* en un ambiente que no puede controlar el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba beta y los reporta al desarrollador periódicamente. Como

resultado de los problemas reportados durante la prueba beta, es posible hacer modificaciones y luego preparar la liberación del producto de *software* a toda la base de clientes.

3.5. Conclusiones parciales

En el presente capítulo se realizó el diagrama de despliegue el cual logró especificar la plataforma sobre la que se ejecuta el *software* del sistema. La definición de los estándares de codificación permitió mejorar el entendimiento del código perteneciente al componente GAMSW 2.0 del Departamento de Componentes del Centro telemática y alcanzar una uniformidad en el mismo. Trazando la estrategia de prueba se definió una serie de pasos bien planificados que dan como resultado una correcta construcción del sistema, entre los que se incluyen los tipos de prueba que se le realizaron al *software*, así como sus niveles. La aplicación del método de caja negra, permitió validar correctamente los requisitos funcionales del sistema utilizándose para ello la técnica de partición de equivalencia. Además, el método de caja blanca contribuyó a validar el código fuente del componente a través de los casos de pruebas usando la técnica de ruta básica. Sin embargo, la realización de la prueba de aceptación demostró que el usuario final está de acuerdo con el componente GAMSW 2.0.

Conclusiones generales

Una vez culminado el presente trabajo de diploma donde se comprende la configuración de servicios web Apache en el Departamento de Componentes del Centro Telemática, se puede concluir que se le dio cumplimiento al objetivo general de la investigación, resaltando que:

1. La elaboración del marco teórico conceptual de la investigación permitió sentar las bases de la misma. Se realizó un estudio de soluciones en el mundo que se relacionan con el proceso de configuración de servicios web, aportando ideas a la aplicación que posteriormente fue desarrollada.

2. El análisis y diseño empleando la metodología AUP-UCI permitió identificar los requisitos de *software* y la descripción de las Historias de Usuario de la propuesta de solución para un mejor detalle de los procesos y funcionalidades de la aplicación desarrollada.

3. La implementación de la solución permitió al sistema GAMSW gestionar de forma centralizada las configuraciones de los servicios web Apache del Departamento de Componentes del centro TLM.

4. Las pruebas permitieron detectar y corregir los errores durante la implementación, posibilitando cumplir con las especificaciones requeridas y la validación de la aplicación implementada.

Referencias Bibliográficas

APACHEGUI, 2015. ApacheGUI. APACHEGUI, 2015. Apache Apache Web Server GUI. 4 de ferero 2015 [en línea]. [Consulta: 20 enero 2019]. Disponible en: https://www.apachelounge.com/viewtopic.php?p=37308.

BOOTSTRAP, 2015. Apuntes de Programación. mayo 4, 2015 [Online] mayo 4, 2015. [Cited: 11 de enero, 2019.] http://programacion.jias.es/2015/05/web-%C2%BFque-es-el-framework-bootstrap-ventajas-desventajas/.

CARRILLO PEREZ, RODRIGO PEREZ GONZALES, DAVID RODRIGUEZ. 2008. METODOLOGÍA DE DESARROLLO DE SOFTWARE. [EN LÍNEA] 2008. [CITADO EL: MAYO 5, 2019.] http://es.slideshare.net/alexandervilcapazachavez/metodologias-de- desarrollo1.

CASTILLO, A.A., 2017. Curso de Programacion Web JavaScript, AJAX y JQuery. CASTILLO, A.A., 2017. Curso de Programacion Web JavaScript, AJAX y JQuery.

CHIU, C.C., ALBERTO, I. y CARBAJAL, T., 2015. Las Pruebas En El Desarrollo De Software. , pp. 46.

COBO, A.Y., 2007. Diseño y programación de bases de datos [en línea]. Madrid. ISBN 9788498214598. Disponible en: https://diatearly-2dcb6.firebaseapp.com/7/DiseÑO-Y-Programacion-De-Base-De-Datos.pdf.

CODEASITE, 2018. HOW DO I FIND APACHE HTTP SERVER LOG FILES. ENERO 2018 [EN LÍNEA]. [CONSULTA: 14 NOVIEMBRE 2018]. Disponible en: https://blog.codeasite.com/how-do-i-find-apache-http-server-log-files/.

CONTRERAS ORTIZ, F.A., 2017. *Dispositivos moviles que permita el pago de servicios publicos de manera segura y confiable*. BOGOTA: 2017.

Django. 2016. BBVA Open4U. [En línea] 2016 de enero de 2016. [Citado el: 11 de mayo de 2019.]

https://bbvaopen4u.com/es/actualidad/django-guia-rapida-para-desarrollar-paginas-web-con-este-framework. GAUCHAT, J., 2012. El gran libro de HTML5, CSS3 y Javascript. Barcelona: Marcombo. ISBN 978- 84-267-1782-5.

GROSSO, A., 2011. Patrones GRASP. ,

HERNÁNDEZ DOMINGUEZ, A., 2018. Investigación y desarrollo. .

HERNÁNDEZ LEÓN, R.A. y COELLO GONZÁLEZ, S., 2012. *El proceso de investigación científica (2a. ed.)*. Rolando Al. Sayda Coello González: s.n. ISBN 8430907114.

IDE, 2015. Joomla! Documentation. *agosto 25, 2015 IDE. 2015. Joomla! Documentation. [En línea] 25 de agosto de 2015. [Citado el: 1 de enero del 2019.] https://docs.joomla.org/Category:IDE_(Integrated_development_environment)/es.*

IVAR JACOBSON, 2000. El Proceso Unificado de Desarrollo de Software (Spanish Edition). Addison, W. Madrid: s.n. ISBN 9788478290369.

INFANTE MONTERO, SERGIO. 2012. MAESTROS DEL WEB. EUGENIA TOBAR. 2012.

JETBRAINS. 2017. PYCHARM. [Online] 2017. [Cited: diciembre 12, 2018.] https://www.jetbrains.com/pycharm/features/.

JM Pantoja Blyde, A Lozano Leal. 2013. Télématique. [Online] 2013. [Cited: octubre 10, 2018.]

http://publicaciones.urbe.edu/index.php/telematique/article/view/2306/pdf.

KABIR, M.J., 2004. La_Biblia_Server_apache. [en línea], vol. 1, pp. 606. Disponible en: https://yexia.files.wordpress.com/2010/09/mohammed-j-kabir-la-biblia-del-servidor-apache-21.pdf.

LARMAN, C., 2003. *UML y Patrones*. segunda. Madrid: s.n. ISBN 0130925691.

MÁRQUEZ, F. y ROSA, O., 2016. La UCI y la informatización de la sociedad cubana. *Animal Genetics*.

MONSALVE GALEANO, E., 2015. PROGRAMA DE INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES. ,

MORA, R.C., 2007. Patrones de GRASP. Adictos al trabajo [en línea]. [Consulta: 24 febrero 2019]. Disponible en: http://www.adictosaltrabajo.com/tutoriales/grasp/.

PARADIGM, V., 2014. Guión Visual Paradigm for UML Índice. [en línea], Disponible en: http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

PÉREZ RÍOS, C.K., 2014. La calidad del servicio al cliente y su influencia en los resultados económicos y financieros de la empresa Restaurante Campestre SAC, Chiclayo periodo enero a setiembre 2011 y 2012. *UNIVERSIDAD CATÓLICA POPULAR DEL RISARALDA*,

PRESSMAN, R.S., 2004. *Ingeniería del software. Un enfoque práctico*. ISBN 0196-2892 VO - 42.

PRESSMAN, R.S., 2010. *Ingeniería del software*. séptima ed. Mexico. ISBN 9786071503145.

RODRÍGUEZ, T., 2014. Metodología de desarrollo para la Actividad productiva de la UCI. , pp. 1-16.

ROSSUM, G., 2017. Tutorial de Python. RODRÍGUEZ, T., 2014. Metodología de desarrollo para la Actividad productiva de la UCI. , pp. 1-16.

SQLITE, 2018. SQlite. [Online] , 2018. [Cited: diciembre 24, 2018.] https://www.sqlite.org.

UBUNTU, 2016. HTTPD - Apache2 Web Server. 15 de mayo del 2015 [en línea]. [Consulta: 12 noviembre 2018]. Disponible en: https://help.ubuntu.com/lts/serverguide/httpd.html.

VILA ALVAREZ, A., 2018. DESARROLLO DE UNA PLATAFORMA WEB PARA LA GESTIÓN DE CONTRATOS DE PRESTACIÓN DE SERVICIOS Y PAZ Y SALVOS PARA EL SERVICIO NACIONAL DE APRENDIZAJE SENA. 2018. S.l.: s.n.

WAHID, H. y AHMAD, S., 2017. Fundamentos de Ingeniería de Software. Jurnal Ekonomi Malaysia, vol. 51, no. 2, pp. 39-54. ISSN 01261962. DOI 10.1017/CBO9781107415324.004.

Webmin. 2016. Webmin. [Online] 2016. [Cited: octubre 24, 2018.] http://www.webmin.com/.

Bibliografía

SOCKETS, C., 2012. Advanced Networks Fall 2012 • Assigned task. [en línea], Disponible en: http://penguin.ewu.edu/cscd433/CourseNotes/CSCD433-Lecture16-2012-Fall-Raw-vs-Cooked-Sockets.pdf.

ALEJANDRO, B.N., JUAN, C.F., IGNACIO, G.C. y EZEQUIEL, Q.M., 1981. *Arquitectura Cliente Servidor*. 1981.

DOWNEY, A.B., ELKNER, J. y MEYERS, C., 2002. Como Pensar como un Científico de la Computación con Python. [en línea], pp. 336. Disponible en: http://www.thinkpython.com.

JANERT, P.K., 2011. *Data Analysis with open Source tools* [en línea]. ISBN 9780596802356. Disponible en: https://www.pdf-archive.com/2012/07/29/oreilly-data-analysis-with-open-source-tools-nov-2010/oreilly-data-analysis-with-open-source-tools-nov-2010.pdf.

PFISTER, G. y POPESCU, D., 1975. Die strenge Approximationseigenschaft lokaler Ringe. *Inventiones Mathematicae*, vol. 30, no. 2, pp. 145-174. ISSN 0020-9910. DOI 10.1007/bf01425506.

WARD, G., 2000. Distribución de módulos Python. [en línea], Disponible en: https://docs.python.org/3/distutils/index.html.

GORALSKI, W., 2017. Dynamic Host Configuration Protocol. *The Illustrated Network*.

WARSAW, B., 2013. Guía de estilo para el código Python – PEP 8 en Español. [en línea], Disponible en: http://www.seleniumhq.org/docs/01_introducing_selenium.jsp.

BARRY, P., 2011. Head First Python - A brain-friendly guide. *Technology*, pp. 1-494.

MICROSOFT, 2016. Insertar, eliminar o cambiar saltos de sección - Word for Mac. *Insertar, eliminar o cambiar saltos de sección* [en línea]. [Consulta: 10 mayo 2019]. Disponible en: https://support.office.com/es-es/article/Insertar-eliminar-o-cambiar-saltos-de-sección-0eeae2d6-b906-42d3-a1bd-7e77ca8ea1f3.

CID, R.V., 2010. Instalación de Servidor Web Apache en Debian - Manuais Informática - IES San Clemente. [en línea]. [Consulta: 14 marzo 2019]. Disponible en: http://informatica.iessanclemente.net/manuais/index.php/Instalación_de_Servidor_Web_Apache_en_Debian.

GEIER, M., 2015. *Introducción a la Computación Python avanzado* [en línea]. ISBN 0123456789. Disponible en: https://docplayer.es/4729468-Introduccion-a-la-computacion-python-avanzado-maximiliano-geier-facultad-de-ciencias-exactas-y-naturales-uba-24-06-2015.html.

BECERRA SANDOVAL, A., 2009. *Introducción a la programación con Python*. ISBN 978-958-8347-22-6.

GRACIA, I. y MARZAL, A., 2003. Introducción a la programación con Python.

HUTCHENS, J. y HIXON, M., 2014. *Kali Linux Network Scanning Cookbook*. ISBN 9781783982141.

KABIR, M.J., 2005. *La Biblia Server Apache* [en línea]. Disponible en: https://yexia.files.wordpress.com/2010/09/mohammed-j-kabir-la-biblia-del-servidor-apache-21.pdf.

COVANTEC, R.L. y MARA, S.C. De, 2016. Materiales del curso de programación en Python - Nivel básico.

RODRÍGUEZ, T., 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2014.

DUQUE, R.G., 2014. Para todos. [en línea], no. 6, pp. 9. Disponible en: http://www.archivogeneral.gov.co/sites/all/themes/nevia/PDF/Transparencia/ACUERDO_02_DE_2014.pdf.

ALVARADO, Y.L., 2016. Proceso de aseguramiento de la calidad para un modelo de la calidad en Cuba. *Revista Cubana de* [en línea], vol. 10, no. Especial Informática 2016, pp. 124-137. Disponible en: http://rcci.uci.cu/index.php?journal=rcci&page=article&op=download&path%5B%5D=1371&path%5B%5D=438.

AKIN, O., LEE, K.J., AKCAMETE, A., AKINCI, B. y GARRETT, J.J., 2009. Product and Process Modeling for Functional Performance Testing in Low-Energy Building Embedded Commissioning Cas. . S.l.: s.n., pp. 1-11. ISBN 0000000000000. DOI 10.1016/S0022-5223(13)00065-2.

SARKER, D.M.O.F., 2014. Python Network Programming. [en línea], no. C, pp. 234. Disponible en: http://it-ebooks.info/book/3515.

ALSINA, R., 2015. *Python no Muerde* [en línea]. S.l.: s.n. Disponible en: http://nomuerde.ralsina.me/python_no_muerde.pdf.

SEVERANCE, C., 2009. Python para inform ´ aticos. [en línea], Disponible en: http://do1.dr-chuck.net/py4inf/ES-es/book.pdf.

MOHIT RAJ, 2015. *Python Penetration Testing Essentials: Employ the power of Python to get the best out of pentesting* [en línea]. S.l.: s.n. ISBN 9781784398583. Disponible en: www.packtpub.com.

ENGINEERS, R., 2015. *Python Programing for Hackers*. S.l.: s.n. ISBN 9781593271923.

LEE, W., 2019. *Python® Machine Learning*. S.l.: s.n. ISBN 9781783555130.

LANGE, W., 2013. Quantum computing with trapped ions. *Computational Complexity: Theory, Techniques, and Applications* [en línea]. S.l.: s.n., pp. 2406-2436. ISBN 9781461418009. Disponible en: https://docs.python.org/3/library/.

CLAVIJO, G., BENITEZ, J., WALLINGRE, O. y WALLINGRE, J., 2009. Redireccionar a otro servidor utilizando una sola IP con mod _ proxy. , pp. 1-10.

THE APACHE *SOFTWARE* FOUNDATION, 2013. Security Tips - Apache HTTP Server. [en línea]. Disponible en: http://httpd.apache.org/docs/2.4/misc/security_tips.html.

UCI, 2013. Servidor web (IIS). [en línea]. [Consulta: 10 mayo 2019]. Disponible en: https://technet.microsoft.com/es-es/library/cc753433(v=ws.10).aspx.

DISTRITAL, U., JOSÉ, F. y FACULTAD, D.C., 1995. SERVIDOR WEB APACHE. , pp. 1-5.

CONWAY, D. y WHITE, J.M., [sin fecha]. Summary for Policymakers. *Climate Change 2013 - The Physical Science Basis* [en línea]. S.l.: s.n., pp. 1-30. ISBN 9788578110796. Disponible en: https://www.cambridge.org/core/product/identifier/CBO9781107415324A009/MaxKeepAliveRequests/book_part.

ERNESTO, I. y VÁZQUEZ, A., 2017. Tema 4 : Programación Orientada a Objetos . Actividad # 9 : Definición del trabajo con clases y objetos Autores. , pp. 1-12.

ROSSUM, G. Van y DRAKE, F.L., 2019. The Python Library Reference - Release 3.7.2. [en línea], pp. 1-1144. ISSN 0169-118X. Disponible en: https://docs.python.org/3/library/.

MULTIPLES, 2010. *Violent Python: A Cookbook for Hackers*. S.l.: s.n. ISBN 9781597499576.

MCKINNEY, W., 2015. *Wes McKinney-Python for Data Analysis-O'Reilly Media (2012)*. S.l.: s.n. ISBN 9781449319793.

ALICATE, U. de, 2013. Arquitectura Cliente Servidor. *Arquitectura Ciiente Servidor* [en línea]. Disponible en: http://www.ecured.cu/index.php/Arquitectura_Cliente_Servidor.

TAHUITON, M.J., 2011. *Arquitectura de software para aplicaciones Web* [en línea]. S.l.: s.n. Disponible en: http://delta.cs.cinvestav.mx/~pmalvarez/tesis-tahuiton.pdf.

CAMACHO, E., CARDESO, F. y NUÑEZ, G., 2004. Arquitecturas de *Software*. *Guía de estudio.[En línea]*, pp. 1-58. Disponible en: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia Arquitectura v.2.pdf%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Arquitecturas+de+*Software*#2.

BERGER, M., 2005. Data Access Object Pattern The Data Access Ob- ject Pattern. , pp. 1-7.

INTRODUCTION, A. y SCHEMA, D.B., 2015. *Data Access Objects (DAOs) An Introduction to DAOs , DB Schema , and SQL*.

TRELLINI, L.A., 2015. Estilos y Patrones Arquitectónicos. *Departamento de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur*, pp. 1-5.

REYNOSO, C.B. y KICILLOF, N., 2004. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. *Universidad de Buenos Aires* [en línea], pp. 28-30. Disponible en: http://carlosreynoso.com.ar/archivos/arquitectura/Estilos.PDF.

WAHID, H. y AHMAD, S., 2017. Fundamentos de Ingeniería de *Software*. *Jurnal Ekonomi Malaysia*, vol. 51, no. 2, pp. 39-54. ISSN 01261962. DOI 10.1017/CBO9781107415324.004.

GROSSMAN, S.L., 2009. *Libros y Soluciones. NET* [en línea]. S.l.: s.n. ISBN 9780521191333. Disponible en: http://librosysolucionarios.net/.

GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E., 2013. Patrones de diseño GOF (the gang of four) en el contexto de procesos de desarrollo de aplicaciones orientadas a la web. *Informacion Tecnologica*, vol. 24, no. 3, pp. 103-114. ISSN 07168756. DOI 10.4067/S0718-07642013000300012.

KUO, B.C., 2000. *Sistemas De Control Automático* [en línea]. ISBN 9688807230. Disponible en: https://www.sistemamid.com/panel/uploads/biblioteca/2014-09-15_01-22-09109838.pdf.

TRELLINI, L.A., 2015. Validación de Arquitectura. , pp. 1-7.

CURTIN, S.J., MICHNO, J.-M., CAMPBELL, B.W., GIL-HUMANES, J., MATHIONI, S.M., HAMMOND, R., GUTIERREZ-GONZALEZ, J.J., DONOHUE, R.C., KANTAR, M.B., EAMENS, A.L., MEYERS, B.C., VOYTAS, D.F. y STUPAR, R.M., 2016. MicroRNA Maturation and MicroRNA Target Gene Expression Regulation Are Severely Disrupted in Soybean dicer-like1 Double Mutants . *G3: Genes|Genomes|GeneTIC* [en línea], vol. 6, no. 2, pp. 423-433. ISSN 0032082X. DOI 10.1534/g3.115.022137. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

AUP Ingeniería de Software [en línea], 2016. 2016. Disponible en: http://ingenieriade*software*.mex.tl/63758_AUP.html.

CALLEJA, M.A., 2010. Carmen. Estándares de codificación. [en línea], Disponible en: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf.

WILLIAMS, J.M.G.., 2010. E n g u a j e s. [en línea], Disponible en: http://www.iqcelaya.itc.mx/~vicente/Programacion/Lenguajes.pdf.

BOOCH, G., RUMBAUGH, J. y JACOBSON, I., 1999. El Lenguaje Unificado de Modelado. *Elements* [en línea], pp. 30. ISSN 1794-1237. DOI 1852 - 4516. Disponible en: http://portal.acm.org/citation.cfm?id=993859&dl=.

6AGUILAR, Y.D. y MOREJÓN, A.R.M., 2013. Gestión de tipos documentales del Sistema para la obtención de documentos digitales con valor legal. [en línea], vol. x, no. x, pp. 1-10. Disponible en: https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjAsp-lvLHbAhVRnFkKHaSxASsQFggoMAA&url=http%3A%2F%2Fdocplayer.es%2F19796903-Yunior-duque-aguilar-1-angel-rolando-maure-morejon-2.html&usg=AOvVaw1INOgYBIfLgkNvVbkYv.

PARADIGM, V., 2014. Guión Visual Paradigm for UML Índice. [en línea], Disponible en: http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

RUMBAUGH, J. JACOBSON, I. &Booch G., 1999. *LenguajeUnificadoModelado.pdf*. 1999.

LÓPEZ, P., 2017. Práctica 1 Herramienta CASE Visual Paradigm Univ . Cantabria – Fac . de Ciencias Visual Paradigm for UML. [en línea], Disponible en: https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwje7pe1tLHbAhVCi1kKHeFtDYcQFgglMAA&url=https%3A%2F%2Focw.unican.es%2Fpluginfile.php%2F1403%2Fcourse%2Fsection%2F1793%2Fis1-p02-trans.pdf&usg=AOvVaw3u5eyqX_2x0etlDx.

ROMEU, I.D., 2016. Sistema Informático De Gestión De Indicadores Para La Formación De Estudiantes Potencialmente Talentosos En La Uci. pp. 1-18.

BLANCO, C., 2014. Objetivos Verificación y Validación. [en línea], pp. 1-85. Disponible en: http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-*software*-ii/materiales/tema1-pruebasSistemas*Software*.pdf.

BEAZLEY, D., 2014. Python Network Programming. [en línea], no. C, pp. 234. Disponible en: http://www.dabeaz.com.

PRESSMAN, R.S., 2010. Ingeniería del Software. Un Enfoque Práctico. [en línea]. SÉPTIMA ED. México: 2010. ISBN 0077096770. Disponible en: http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.

NERI, A.B., FALCON, J.C., CASTILLO, I.G. y MARTINEZ, E.Q., 2018. Arquitectura Cliente Servidor. [en línea], vol. 3, no. Septiembre, pp. https://www.arduino.cc/en/Guide/Introduction. Disponible en: http://profesores.fi-b.unam.mx/yasmine/expo/Sockets_Phyton.pdf.

COVANTEC, R.L. y MARA, S.C. De, 2016. Materiales del curso de programación en Python - Nivel básico. [en línea], vol. 1. Disponible en: https://www.lawebdelprogramador.com/pdf/15299-Materiales-del-curso-de-programacion-en-Python.html.

VAN ROSSUM, G. y DRAKE, F.L., 2005. Guía de aprendizaje de Python.

SEITZ, J., 2013. Summary for Policymakers. Climate Change 2013 - The Physical Science Basis [en línea]. S.l.: s.n., pp. 1-30. ISBN 9788578110796. Disponible en: https://www.cambridge.org/core/product/identifier/CBO9781107415324A009/type/book_part.

LEE, W., 2019. Python® Machine Learning. ISBN 9781783555130.

MCKINNEY, W., 2015. Summary for Policymakers. Climate Change 2013 - The Physical Science Basis [en línea]. pp. 1-30. ISBN 9781449319793. Disponible en: https://www.cambridge.org/core/product/identifier/CBO9781107415324A009/type/book_part.

CERIA, S., 2001. Casos de uso: Un Método Práctico para Explorar Requerimientos. http://www-2.dc.uba.ar/materias/isoft1/2001_2/apuntes/CasosDeUso.pdf. [en línea], vol. 1, pp. 18. Disponible en: http://www-2.dc.uba.ar/materias/isoft1/2001_2/apuntes/CasosDeUso.pdf.

GARCÍA CASADO, M. del M., 2017. Mendeley y APA. Cómo utilizar Mendeley para redactar la bibliografía en formato APA 6th. Manuales [en línea]. S.l.: Disponible en: http://hdl.handle.net/10612/5132.

TRELLINI, L.A., 2015. Estilos y Patrones Arquitectónicos. Departamento de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur, pp. 1-5.

Anexos

Anexo 1: Historia de Usuario

Tabla 10: Descripción del RF 3

Historia de Usuario	
Número: 3	**Nombre del Requisito:** Modificar archivo principal
Prioridad: Alta	
Riesgo en Desarrollo: Afectaciones al personal de trabajo debido a orientaciones de la dirección o la universidad, enfermedad y otros factores.	
Descripción: ✓ Permite al administrador realizar modificaciones a algunos parámetros del archivo principal de Apache.	
Observaciones: ✓ El archivo principal de Apache puede ser httpd.conf o apache2.conf, dependiendo de la distribución con la que esté trabajando el especialista diestro en Linux. En el presente caso es de interés el archivo apache2.conf.	
Prototipo de Interfaz	

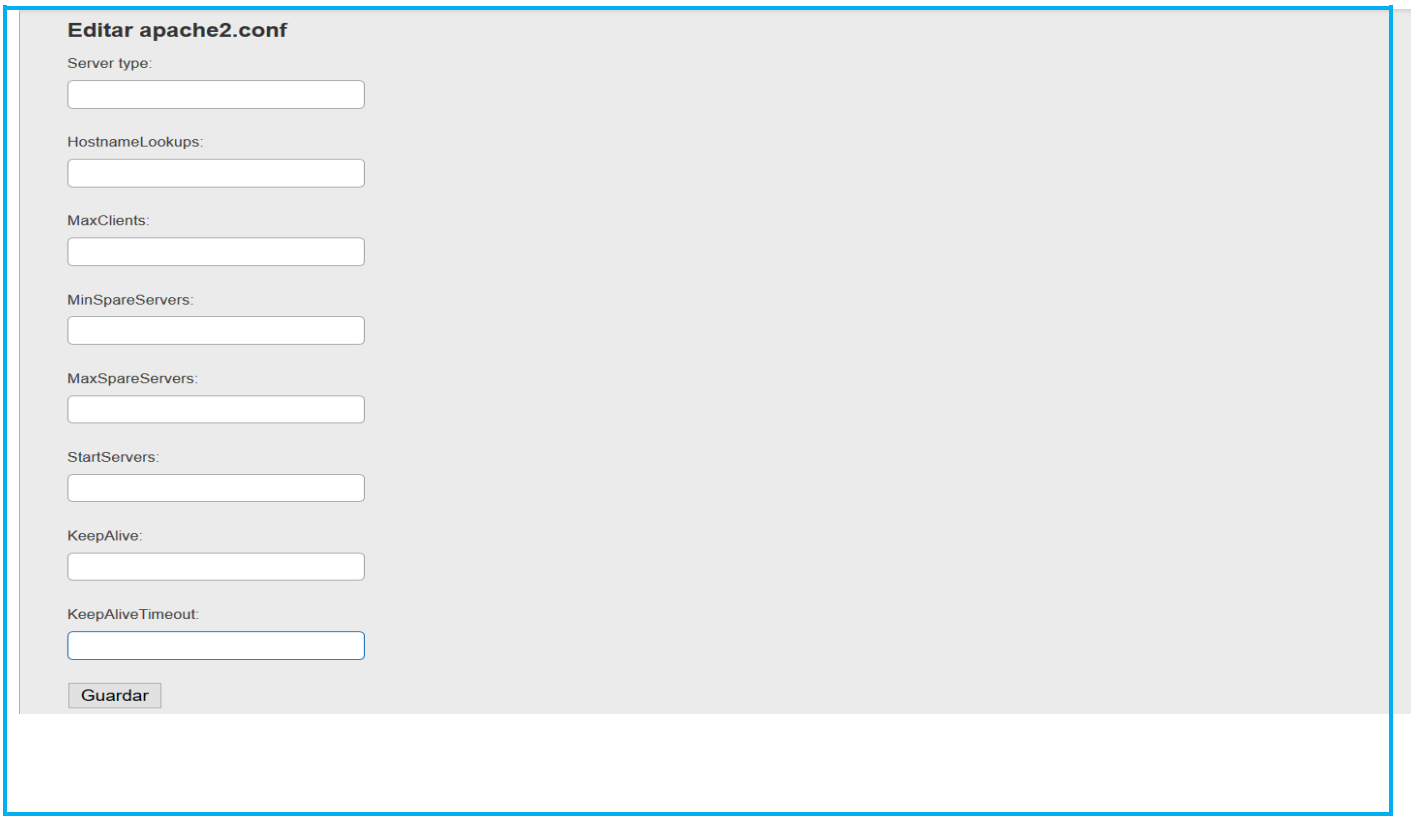


Tabla 11: Descripción del RF 3.1

Historia de Usuario	
Número: 3.1	**Nombre del Requisito:** Mostrar parámetros configurables
Prioridad: Alta	
Riesgo en Desarrollo: Afectaciones al personal de trabajo debido a orientaciones de la dirección o la universidad, enfermedad y otros factores.	
Descripción: ✓ El sistema accede al servidor remoto a través del protocolo SSH y busca el archivo apache2.conf en el directorio var/apache2/. ✓ Al tener acceso al contenido del archivo muestra un listado con los parámetros: server MaxKeepAliveRequests, Hostnamelookups, ServerRoot: , ErrorLog, LogLevel, Timeout, KeepAlive,	

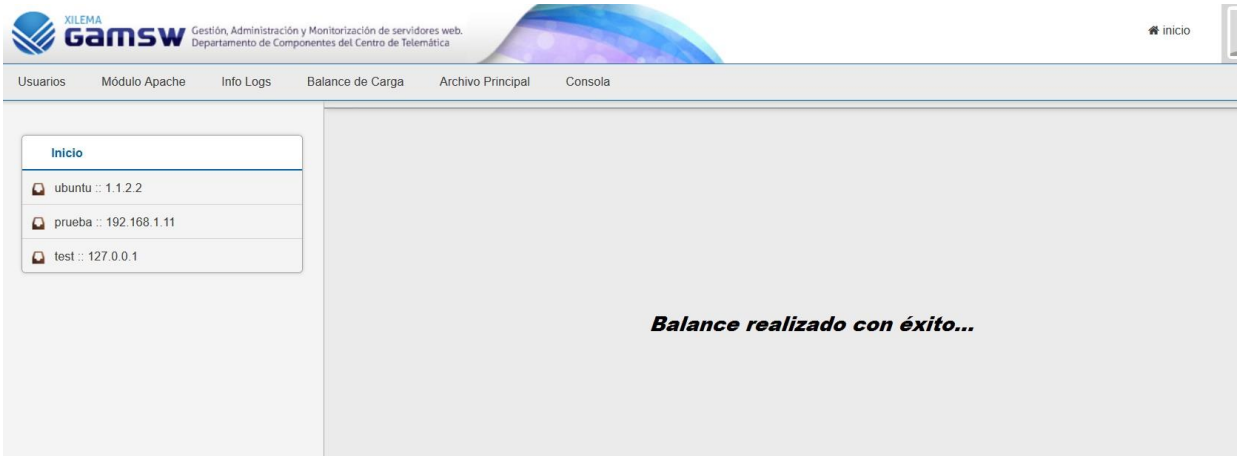
KeepAlive Timeout. ✓ El administrador podrá seleccionar el parámetro que desee configurar
Observaciones: ✓ El administrador tiene acceso a los parámetros de configuración de manera gráfica.

Tabla 12: Descripción del RF 3.2

Historia de Usuario	
Número: 3.2	**Nombre del Requisito:** Modificar parámetro
Prioridad: Alta	
Riesgo en Desarrollo: Afectaciones al personal de trabajo debido a orientaciones de la dirección o la universidad, enfermedad y otros factores.	
Descripción: ✓ Permite al administrador realizar configuraciones en un parámetro seleccionado y dar clic en el botón Guardar. ✓ El sistema validará los cambios realizados en el formulario, si están correctos guardará la configuración en el archivo principal, en caso contrario mostrará un mensaje de error.	
Observaciones: ✓ Al archivo al ser modificado guardará su nueva configuración y podrá ser modificado posteriormente.	

Tabla 13: Descripción del RF 4

Historia de Usuario	
Número: 4	**Nombre del Requisito:** Realizar Balanceo de Carga
Prioridad: Alta	
Riesgo en Desarrollo: Afectaciones al personal de trabajo debido a orientaciones de la dirección o la universidad, enfermedad y otros factores.	

Descripción: ✓ Permite al administrador realizar un balanceo de carga entre los distintos servidores web Apache cuando lo considere conveniente.	
Observaciones: ✓ Al realizarse el balanceo de carga, las peticiones que se encuentran alojadas en los servidores más saturados serán repartidas entre otros servidores con bajo nivel de peticiones para atender.	
Prototipo de Interfaz 	

Anexo 2: Prueba de caja negra

Tabla 14: Prueba de caja negra del RF Mostrar Información de log

Escenario	Descripción	Ruta visitada	IP visitante	Horario Visita	Navegador	Sistema Operativo	Estilo gráfico	Respuesta del Sistema	Flujo Central
EC 1.1 Buscar Log	El sistema busca en el servidor web Apache el log que contiene la	N/A	N/A	N/A	N/A	N/A	N/A	Valida la selección del log	Seleccion a el log contenedo r de la informació

	información de interés								n de interés
EC 1.2 Buscar Parámetros	El sistema busca en el log los parámetros de interés ubicadas en posiciones específicas	N/A	N/A	N/A	N/A	N/A	N/A	Valida la selección de los parámetros	Seleccion a los parámetro s de interés ara ser mostrados
EC 1.3 Mostrar parámetros	Permite mostrar los parámetros de interés	V	V	V	V	V	V	Muestra los parámetros en una tabla	1-Inserta los parámetro s en una tabla. 2-Seleccion a la interfaz para mostrar la tabla
		V	V	V	V	V	I	No muestra los parámetros en un estilo gráfico acorde al del sistema	3-Seleccion a un estilo gráfico para los parámetro s